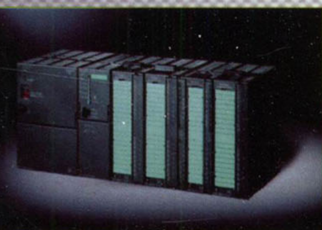
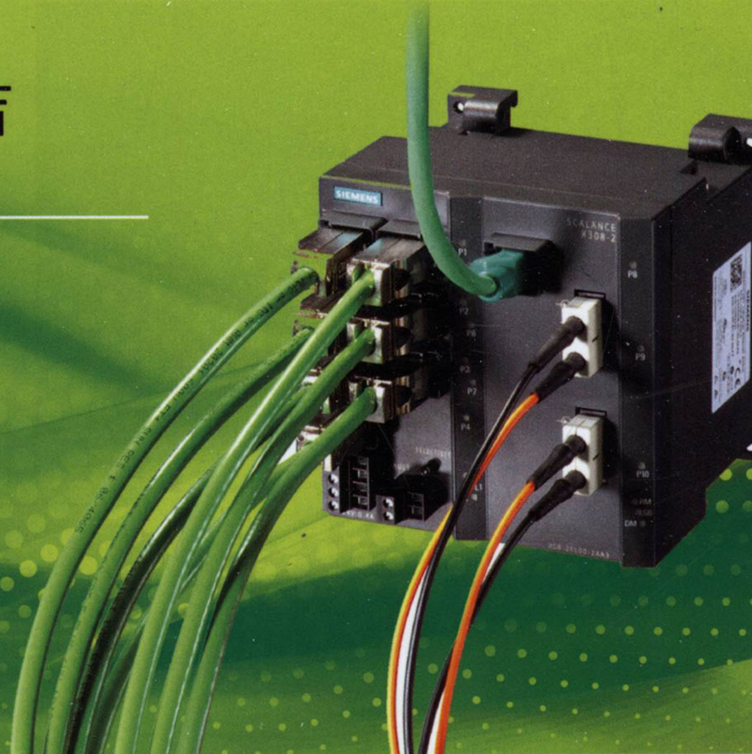
西门子PLC 控制技术



主编　王德吉





机械工业出版社
CHINA MACHINE PRESS

VISIT…

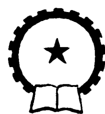
LANZAROTE Caliente.COM



作者简介：王德吉，中共党员，中国烟草总公司职工进修学院首席培训师，博士生导师。从事复杂烟草自动化与信息化研究与培训工作。兼职国家高级考评员，中科院高级访问学者，湖南农业大学学术型硕士生导师，清华大学工程硕士研究生企业导师，郑州大学兼职教授，国家管道工程实验室特聘专家，全国工业过程测量和控制标准化技术委员会委员，工业物流自动化河南省工程实验室副主任，中国自动化学会委员，国际自动控制联合会委员，国际工程师协会委员，中国创新方法研究会首批会员，西门子自动化专家，中国科学技术协会企业创新专家。先后获省部级奖一等奖2项，软件著作权6项，著作2项，发明专利3项，发表论文30余篇，其中SCI、EI检索19篇。主持省部级重大专项1项，参与国家863项目3项，国家自然科学基金项目1项，国家973项目1项，主持企业合作项目6项。先后在全国创新技术大会、海峡两岸创新方法研讨会、博鳌论坛、国际自动化会议、国际工程师大会、中德职业教育国际交流大会发言交流，先后访问德国费斯托公司、德国虹霓公司和英国埃塞克斯大学和牛津大学，先后接受《东方烟草报》、安徽电视台、《自动化博览》等杂志采访报道。

西门子 PLC 控制技术

主　编　王德吉
副主编　黄光富　陈智勇



机 械 工 业 出 版 社

本书以西门子公司 S7-300 PLC 为主要介绍对象，以 PLC 的应用技术为重点，淡化原理，注重实用，以项目、实例为线索进行内容的编排，介绍了 PLC 的工作原理、内部存储区、指令系统、程序结构、编程软件的使用、编程规则与技巧、控制系统设计与应用技术等。

全书语言简洁、通俗易懂、内容丰富、实用性强、理论联系实际。本书可作为电气自动化、自动化、楼宇自动化、机电一体化、机械设计与制造及其相关专业 PLC 应用系统设计与安装课程的教学用书，也可作为电气技术人员的参考书和培训教材。

本书课件请在 http://www.cmpbook.com/index.php? id=134 下载。

图书在版编目（CIP）数据

西门子 PLC 控制技术/王德吉主编. —北京：机械工业出版社，2014.3
ISBN 978-7-111-46052-7

Ⅰ.①西…　Ⅱ.①王…　Ⅲ.①plc 技术　Ⅳ.①TM571.6

中国版本图书馆 CIP 数据核字（2014）第 040384 号

机械工业出版社（北京市百万庄大街 22 号　邮政编码 100037）
策划编辑：林春泉　责任编辑：赵　任　版式设计：常天培
责任校对：刘志文　封面设计：路恩中　责任印制：乔　宇
北京机工印刷厂印刷（三河市南杨庄国丰装订厂装订）
2014 年 6 月第 1 版第 1 次印刷
184mm×260mm · 27.25 印张 · 737 千字
标准书号：ISBN 978-7-111-46052-7
定价：78.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务	网络服务
社服务中心：（010）88361066	教材网：http://www.cmpedu.com
销售一部：（010）68326294	机工官网：http://www.cmpbook.com
销售二部：（010）88379649	机工官博：http://weibo.com/cmp1952
读者购书热线：（010）88379203	**封面无防伪标均为盗版**

序

随着微处理器技术、电力电子技术、网络技术和控制技术的发展，PLC 系统实现了全数字化、智能化、网络化，已成为自动控制系统发展的主流方向。PLC 控制技术在先进制造领域以及在提高生产速度、管理生产过程、合理高效加工和保证安全生产等方面起到越来越关键的作用。

烟草行业是全球各生产制造行业中自动化程度比较高的行业之一。世界各地的许多卷烟厂都热衷于自动化技术改造，无论是欧美发达国家，还是亚洲新兴工业化国家的卷烟生产企业，在自动化技术改造方面都不遗余力。中国烟草总公司从 20 世纪 80 年代初期引进国外先进的生产设备开始，就在不断地探索和追踪最新的自动化技术。目前，烟草企业对学习西门子 PLC 技术的呼声越来越强烈。

我院作为中国烟草行业技术人才的专门培训机构，多年来一直致力于追踪全球工业自动化领域最新的技术和理念并将其转化为生产力，服务于国内的烟草企业，并为之培养培训了大批的技术及技术管理人才。近年来，为满足烟草企业针对工业控制方面的特殊培训需求，我院与西门子公司展开了多层次、全方位的人才培训合作。《西门子 PLC 控制技术》一书就是西门子公司与我院机电工程研究室合作的结晶。该书将西门子 PLC 技术汇集其中，并结合现场应用实例，将西门子 PLC 技术详尽地呈现给读者。希望该书能为自动化领域设计人员和企业工程师及院校师生提供有力支持和帮助。由于时间有限、水平有限，书中难免有不妥之处，恳请广大读者提出批评和修正意见。

中国烟草总公司职工进修学院
副院长　李广才
2013 年 10 月

前　　言

当前大中型PLC应用非常广泛，而相关的参考书和教材都比较少，对初涉PLC技术的读者学习应用起来入门难、跨度大。本书从实际应用角度出发，以应用最广泛的西门子公司的S7-300/400系列PLC为对象而编写的是一本独具特色的教材和参考书。

PLC以微处理器为核心，将微型计算机技术、自动控制技术及网络通信技术有机地融为一体，是应用十分广泛的通用工业自动化控制装置。它具有控制能力强、可靠性高、配置灵活、编程简单、使用方便、易于扩展等优点，是当今及今后工业控制的主要手段和重要的自动化控制设备。

西门子公司的S7-300/400在大中型PLC中应用最为广泛，市场占有率最高。S7-300/400及其编程软件STEP 7和通信网络的功能强大，但其程序结构和网络的组建比较复杂，不易掌握。虽然西门子公司为此编写了相应的硬件安装手册、程序编写手册和网络通信连接手册，但对所有类型的PLC的介绍面面俱到没有突出现阶段重点使用的几种类型，并且所有参考手册都是英文版的，这就要求用户具有较高的英语水平，学习起来比较困难。对此，本书弥补了这些缺憾，结合实例，深入浅出地讲述了西门子S7-300/400系列PLC的应用。

本书具有以下特色：具有非常详尽的指令系统说明，书中对所有的指令都进行了详细的介绍，并针对指令列举了相应的编程实例，使学习者能尽快地掌握指令的应用方法；强调实际应用，给出了S7-300/400PLC的一些实用性很强的应用实例，以提高学习者对S7-300/400PLC工程应用的认识。因此，本书既可以作为高等院校相关专业的教材，也可以作为工程技术人员的参考书。

本书由王德吉任主编，黄光富、陈智勇任副主编，参加本书的编写人员有张建勋、张晓峰、李文磊、丁文萍、王玉、栗卫军、杨彬、李源源、张旭、谢俊明、马晓、汪翠兰、王德玉。

由于编者水平有限，书中难免存在缺点和不足，殷切地希望广大读者提出宝贵的意见和建议。

编　者

2013年10月

目　录

序
前言
第一章　可编程序控制器的基础知识 ……… 1
第一节　PLC 概述 ……………………………… 1
一、PLC 的产生与发展 ……………………… 1
二、PLC 的特点 ……………………………… 3
第二节　PLC 的组成 …………………………… 3
第三节　PLC 的工作原理 ……………………… 7
第四节　PLC 的硬件基础 ……………………… 9
一、PLC 的 I/O 模块 ………………………… 9
二、PLC 的配置 ……………………………… 11
第五节　PLC 的软件基础 ……………………… 11
一、系统监控程序 …………………………… 11
二、用户应用程序 …………………………… 12
第六节　PLC 的性能指标及分类 …………… 14
一、按结构形式分类 ………………………… 14
二、按功能分类 ……………………………… 15
三、按 I/O 点数分类 ………………………… 15
第二章　西门子公司常用系统简介 ……… 16
第一节　SIMATIC PLC 控制器 ……………… 16
一、SIMATIC S7-200 ……………………… 16
二、SIMATIC S7-300 ……………………… 16
三、SIMATIC S7-400 ……………………… 17
第二节　工业通信 ……………………………… 17
一、工业以太网 ……………………………… 18
二、现场总线 PROFIBUS …………………… 18
三、AS-i 电缆连接 ………………………… 19
第三节　人机界面 ……………………………… 19
第四节　SIMATIC 工业软件 ………………… 19
一、STEP 7 ………………………………… 19
二、顺序控制编程软件 S7-GRAPH ………… 21
三、状态控制编程软件 S7-HiGRAPH ……… 21
四、高级编程语言 S7-SCL ………………… 21
五、SIMATIC WinAC Basis ………………… 22
六、SIMATIC ProTool/Pro ………………… 23
七、HMI SIMATIC WinCC ………………… 23
八、PCS 7 过程控制系统 …………………… 24
第五节　驱动技术 ……………………………… 24
一、低压电动机 ……………………………… 24
二、SIMOVERT MASTERDRIVES 变频器…… 25
三、标准变频器 ……………………………… 25
四、SIMOREG 直流调速器 ………………… 25
第三章　S7-300/400 PLC 的硬件配置 …… 26
第一节　S7-300 的基本组成 ………………… 26
一、S7-300 的概况 ………………………… 26
二、S7-300 的系统结构 …………………… 26
三、S7-300 模块诊断与过程诊断 ………… 28
第二节　S7-300 的功能模块 ………………… 29
一、S7-300 的 CPU ………………………… 29
二、S7-300 的数字量模块 ………………… 30
三、S7-300 的模拟量模块 ………………… 31
四、S7-300 的电源模块 …………………… 32
五、数字量的 I/O 编址 …………………… 32
六、其他功能模块 …………………………… 33
第三节　S7-400 系统简介 …………………… 33
一、S7-400 的系统结构 …………………… 34
二、S7-400 的优点 ………………………… 34
三、S7-400 的通信功能 …………………… 35
第四节　机架与接口模块 ……………………… 35
一、机架 ……………………………………… 35
二、接口模块 ………………………………… 36
三、错误诊断 ………………………………… 36
四、冗余设计 ………………………………… 37
第五节　S7-300/400 扩展机架的配置与说明 …………………………………… 38
一、S7-300 系统扩展 ……………………… 38
二、S7-400 系统扩展 ……………………… 42
三、组态 ……………………………………… 51
第六节　多 CPU 处理及 CPU 模块 ………… 52
一、多 CPU 处理 …………………………… 52
二、CPU 模块的元件 ……………………… 52
第四章　S7-300/400 PLC 的常用指令…… 54
第一节　S7-300/400 PLC 编程基础 ……… 54
一、编程语言 ………………………………… 54
二、数据类型 ………………………………… 55
三、存储器区域 ……………………………… 57

四、寻址方式 …… 60
五、编程的一般规则 …… 65
第二节 S7-300/400 PLC 的指令系统 …… 65
一、位逻辑指令 …… 66
二、比较指令 …… 73
三、转换指令 …… 75
四、计数器指令 …… 83
五、数据块指令 …… 87
六、逻辑控制指令 …… 89
七、整型数学运算指令 …… 98
八、浮点运算指令 …… 104
九、装载和传送指令 …… 112
十、程序控制指令 …… 117
十一、移位和循环移位指令 …… 130
十二、状态位（LAD）指令 …… 140
十三、定时器指令 …… 144
十四、字逻辑指令 …… 158
十五、累加器（STL）指令 …… 164
第五章 西门子编程软件 STEP 7 …… 168
第一节 STEP 7 编程软件的使用简介 …… 168
一、STEP 7 概述 …… 168
二、STEP 7 标准软件包 …… 168
三、STEP 7 的授权 …… 168
四、STEP 7 的安装和硬件接口 …… 169
五、STEP 7 的编程功能 …… 170
六、STEP 7 的硬件组态与诊断功能 …… 170
第二节 硬件组态与参数设置 …… 171
一、项目的创建与项目的结构 …… 171
二、硬件组态 …… 172
三、CPU 模块的参数设置 …… 175
四、数字量输入模块的参数设置 …… 175
五、数字量输出模块的参数设置 …… 176
六、模拟量输入模块的参数设置 …… 176
七、模拟量输出模块的参数设置 …… 176
第三节 定义符号 …… 177
第四节 创建逻辑块 …… 179
一、块文件 …… 179
二、逻辑块的创建 …… 180
三、程序编辑器窗口的结构 …… 180
四、程序指令输入 …… 181
五、程序下载和上传 …… 183
第五节 仿真软件使用与说明 …… 185
一、与“真正”PLC 的区别 …… 186
二、起动仿真 …… 189
三、S7-PLCSIM 的使用 …… 193
四、故障排除提示 …… 196
第六节 调试 …… 200
一、用变量表调试 …… 200
二、用编程状态调试 …… 203
第七节 故障诊断 …… 206
一、故障诊断的基本方法 …… 206
二、用快速视图和诊断视图诊断故障 …… 207
三、调用模块信息诊断故障 …… 209
第八节 显示参考数据 …… 210
一、参考数据的生成与显示 …… 210
二、交叉参考表 …… 210
三、程序结构 …… 211
四、赋值表 …… 212
五、未使用的符号 …… 212
六、不带符号的地址 …… 213
第六章 S7-300/400 用户程序结构与编程 …… 214
第一节 用户程序的基本结构 …… 214
一、用户程序中的块 …… 214
二、用户程序使用的堆栈 …… 216
三、STEP7 编程方式 …… 217
第二节 功能块与功能的调用 …… 218
一、局域变量的类型 …… 218
二、功能块与功能的调用 …… 218
第三节 数据块 …… 222
一、数据块的生成与使用 …… 222
二、数据块中的数据类型 …… 223
第四节 多重背景 …… 224
一、多重背景功能块的生成 …… 224
二、多重背景功能块的编程 …… 225
三、在 OB1 中调用多重背景 …… 227
第五节 组织块与中断处理 …… 228
一、中断的基本概念 …… 228
二、组织块的变量声明表 …… 229
三、日期时间中断组织块（OB10 ~ OB17） …… 229
四、时间延时中断组织块 …… 230
五、循环中断组织块 …… 230
六、硬件中断组织块 …… 231
七、背景组织块 …… 231
八、起动组织块 OB100/OB101/OB102 …… 232
九、故障处理组织块 …… 233

十、同步错误组织块 …… 234
十一、常用 OB 组织块的使用举例 …… 235
第六节 常用模拟量的处理 …… 258
一、模拟量模块的用途 …… 258
二、模拟量寻址 …… 260
三、模拟输入量的规范化 …… 264
四、模拟量输出的规范化 …… 265
第七节 在 STEP7 中实现 PID 控制 …… 267
一、概述 …… 267
二、PID 系统控制器的选择 …… 271
三、布线 …… 272
四、参数赋值工具介绍 …… 272
五、在用户程序中实现 …… 273
六、功能块介绍 …… 274
七、功能块举例 …… 290
第七章 S7-300/400 的通信及网络 …… 291
第一节 通信及网络基础 …… 291
一、数据通信方式 …… 291
二、信道和信道参数 …… 293
三、传送介质 …… 294
四、网络传输设备 …… 295
第二节 通信网络结构 …… 297
一、网络概述 …… 297
二、网络体系结构——IEEE802 参考模型和 ISO 标准 …… 297
三、数据通信的网络拓扑结构 …… 301
四、现场总线 …… 303
第三节 S7-300/400 的通信网络 …… 304
一、工业自动化网络 …… 304
二、S7-300/400 的通信网络 …… 305
三、通信的分类 …… 307
四、MPI 全局数据通信 …… 307
五、MPI 网络的组建 …… 308
六、MPI 网络组态 …… 310
第四节 PROFIBUS 概述 …… 313
一、PROFIBUS 的组成 …… 313
二、PROFIBUS 的物理层 …… 313
三、PROFIBUS-DP 设备的分类 …… 314
四、PROFIBUS 的通信协议 …… 314
五、基于组态的 PROFIBUS 通信 …… 316
第五节 执行器传感器接口网络 …… 319
一、AS-i 的寻址模式 …… 320
二、AS-i 网络接口部件 …… 320
三、AS-i 主站模块 …… 321
四、从站模块 …… 321
五、AS-i 的主从通信方式 …… 323
六、AS-i 的工作模式 …… 323
第六节 点对点通信 …… 324
一、点对点通信处理器与集成的点对点通信接口 …… 324
二、ASCII Driver 通信协议 …… 325
三、3964（R）通信协议 …… 325
四、用于 CPU 31xC-2PtP 点对点通信的系统功能块 …… 326
第七节 工业以太网 …… 327
一、工业以太网介绍 …… 327
二、工业以太网的网络方案 …… 328
三、工业以太网的交换技术 …… 329
第八章 PLC 工程应用开发 …… 330
第一节 工程设计原则 …… 330
第二节 需求分析 …… 331
第三节 硬件设计 …… 331
一、PLC 机型选择 …… 331
二、确定容量参数 …… 332
三、系统软、硬件选择 …… 333
第四节 软件设计 …… 333
一、控制程序的设计 …… 333
二、控制系统的设计 …… 335
第五节 系统调试 …… 336
第六节 可靠性设计 …… 338
一、影响现场输入给 PLC 信号出错的主要原因 …… 338
二、影响执行机构出错的主要原因 …… 338
三、硬件可靠性设计 …… 338
四、软件可靠性设计 …… 341
第七节 编程实例与工程应用 …… 342
一、简单编程实例 …… 343
二、运料小车控制系统 …… 380
三、水塔水位控制 …… 384
四、四节传送带控制系统 …… 386
五、电梯控制系统 …… 390
六、机械手控制系统线性程序设计 …… 399
第九章 常见故障现象与原因分析 …… 404
第一节 常见故障的检查与处理 …… 404
一、常见故障的总体检查与处理 …… 404
二、电源故障检查与处理 …… 404

三、异常故障检查与处理 …………………… 404
四、通信故障检查与处理 …………………… 405
五、I/O 故障检查与处理…………………… 405
六、定期检修 ……………………………… 406
七、PLC 的故障处理 ……………………… 407
第二节 常见问题及解答 …………………… 407
一、如何将二线制测量传感器连接到模拟量模块、紧凑型 CPU 或 C7 设备 ……… 407
二、S7-300 模拟量输入模块测量温度时的测量误差 ……………………………… 408
三、把一个 PT 100 温度传感器连接到 SM331 ……………………………… 408
四、将 HART 测量传感器连接到常规的 S7-300 模拟输入模块是可行的 ……… 409
五、有关 SM 335 正确接线的信息 ………… 409
六、怎样理解 S7-400 数据的存储及存储容量，如何查找 CPU 的存储器参数 …… 409
七、如何利用 OB81 判断电源故障………… 410
八、如何能在不重新起动系统的情况下，改变 PUT 和 GET SFBs（SFB14、15）的 ID 参数 ……………………………… 411
九、使用系统功能块 SFB12 和 SFB13（BSEND BRCV）时应注意些什么 ……………… 411
十、为什么具有诊断功能的数字输出模块 SM422-7BL 的外部故障灯（EXTF），在清除输出与地短路的情况下仍常亮 … 412
十一、当测量值为“7FFF”时，如何分辨是断线故障还是测量值溢出 ……… 412
十二、为什么尽管插入一块新的备用电池还出现电池故障信号 ……………… 412
十三、何时更换 S7-300/400 控制器的备用电池 ……………………………… 412
十四、当采用交流电源供电时，应该选择哪种电源进线断路器 ……………… 413
十五、当 24V 电源过载时 S7-400 有何反应，电源模块又如何反应 ……………… 413
十六、300 系列以太网 CP 模板有什么不同 ……………………………… 413
十七、SIMATIC S7-300/400 如何使用 BSEND BRCV 确保数据传输的一致性……… 415
十八、哪些 IP 地址与哪些子网掩码相互兼容 ……………………………… 415
十九、如何在 TCP/IP 网络中分配 IP 地址和子网掩码 ……………………… 416
第十章 西门子 PLC 远程访问诊断方案 ……………………………… 418
第一节 基于 Modem 拨号的 TeleService …… 418
第二节 基于互联网的 TeleService ………… 418
一、有线连接方式 ………………………… 418
二、无线方式（CDMA/GPRS）建立 VPN ……………………………………… 425

第一章　可编程序控制器的基础知识

本章内容包括可编程序控制器（Programmable Logic Controller，PLC）产生的背景、特点、组成、发展以及 PLC 工作的一般原理。通过对本章的学习，掌握 PLC 的基础知识，以有利于后面章节内容的学习。

第一节　PLC 概述

PLC 是在电器控制技术和计算机技术的基础上开发出来的，并逐渐发展成为以微处理器为核心，把自动化技术、计算机技术、通信技术融为一体的新型工业控制装置。目前，PLC 已被广泛地应用于各种生产机械和生产过程的自动控制中，成为一种最重要、最普及、应用场合最多的工业控制装置，被公认为现代工业自动化的三大支柱（PLC、机器人、CAD/CAM）之一。

国际电工委员会（IEC）于 1987 年颁布了 PLC 标准草案第三稿，在草案中对 PLC 定义如下："PLC 是一种数字运算操作的电子系统，专为在工业环境下应用而设计。它采用可编程序的存储器，用来在其内部存储执行逻辑运算、顺序控制、定时、计数和算术运算等操作的指令，并通过数字式和模拟式的输入和输出，控制各种类型的机械或生产过程。PLC 及其有关外围设备，都应按易于与工业系统联成一个整体，易于扩充其功能的原则来设计"。

定义强调了 PLC 应直接应用于工业环境，必须具有很强的抗干扰能力、广泛的适应能力和广阔的应用范围，这是区别于一般微机控制系统的重要特征。同时，也强调了 PLC 用软件方式实现的"可编程"与传统控制装置中通过硬件或硬接线的变更来改变程序的本质区别。

近年来，PLC 发展很快，几乎每年都推出不少新系列产品，其功能已远远超出了上述定义的范围。

一、PLC 的产生与发展

在制造业和过程工业中，除了以模拟量为被控对象的反馈控制外，还存在着大量的以开关量（数字量）为主的逻辑顺序控制，这一点在以改变几何形状和机械性能为特征的制造工业中显得尤其突出。它要求控制系统按照逻辑条件和一定的顺序、时序产生控制动作，并能够对来自现场的大量的开关量、脉冲、计时、计数以及模拟量的越限报警等数字信号进行监视和处理。这些工作在早期是由继电器电路来实现的，其缺点是体积庞大、故障率高、功耗大、不易维护、不易改造和升级等。

1968 年，美国通用汽车公司（GM）鉴于传统的继电器控制系统的一系列缺点，提出了研制新型控制器的设想，总结出新型控制器应当具有的 10 项指标，并以此公开在社会上招标，这 10 项指标是：

1）编程方便，可在现场修改程序。

2）维护方便，最好是插件式。

3）可靠性高于继电器控制柜。

4）体积小于继电器控制柜。

5）可将数据直接送入管理计算机。

6）在成本上可与继电器控制柜竞争。

7）输入为交流 115V。

8）输出为交流 115V/2A 以上，能直接驱动电磁阀、接触器等。

9）在扩展时原有系统改变最少。

10）用户程序存储器至少可扩展到 4KB。

美国数字设备公司（DEC）根据这 10 项指标，于 1969 年研制出第一台控制器，型号为 PDP-14，它的开创性意义在于引入了程序控制功能，为计算机技术在工业控制领域的应用开辟了新的空间。

至 20 世纪 70 年代，PLC 技术已经进入成熟期。推动 PLC 技术发展的动力主要来自于两个方面，其一是企业对高性能、高可靠性自动控制系统的客观需要和追求，例如关于 PLC 最初的性能指标就是由用户提出的。其次，大规模及超大规模集成电路技术的飞速发展，微处理器性能的不断提高，为 PLC 技术的发展奠定了基础并开拓了空间。这两个因素的结合，使得当今的 PLC 已经在所有性能上都大大超越了前述的 10 项指标。

现在，PLC 的程序存储容量多以 MB 为单位，随着超大规模集成电路技术的发展，微处理器的性能大幅提高，指令执行速度达到微秒级，从而极大提高了 PLC 的数据处理能力，高档的 PLC 可以进行复杂的浮点数运算，并增加了许多特殊功能，例如高速计数、脉宽调制变换、PID 闭环控制、定位控制等，从而在以模拟量为主的过程控制领域也占有了一席之地，在一定程度上具备了组建 DCS 的能力。此外，PLC 的通信功能和远程 I/O 能力也非常强大，可以组建成分布式通信网络系统。

在组成结构上，PLC 具有一体化结构和模块式结构两种模式。一体化结构的 PLC 追求功能的完善，性能的提高，体积越来越小，有利于安装。而模块式结构，则是利用单一功能的各种模块拼装成一台完整的 PLC，用户在设计自己的 PLC 控制系统时拥有极大的灵活性，并使设备的性价比达到最优。同时，模块式结构也有利于系统的维护、换代和升级，并使系统的扩展能力大大加强。

在控制规模上，PLC 向小型化和大型化两个方向发展。大型 PLC 是基于满足大规模、高性能控制系统的要求而设计的，在规模上，可带的 I/O 点数（通道数量）达到数千点乃至上万点。在对高性能的追求上，主要体现在以下几点：

1）增强网络通信功能。这是 PLC 的一个重要发展趋势，伴随现场总线（Field Bus）技术的应用，由多个 PLC、多个分布式 I/O 模块、人机界面、编程设备相互连接成的网络，与工业计算机和以太网等构成整个工厂的自动控制系统。PLC 采用了计算机信息处理技术、网络通信技术和图形显示技术，使得 PLC 系统的生产控制功能和信息管理功能融为一体。

2）发展智能模块。智能模块以微处理器为核心，与 PLC 的 CPU 并行工作，完成专一功能，大量节省主 CPU 的时间和资源，对提高用户程序的扫描速度和完成特殊的控制要求非常有利。例如通信模块、位置控制模块、模糊逻辑控制模块、高速计数器模块等。

3）高可靠性。PLC 广泛采用自诊断技术，向用户提供故障分析的信息和提示。同时，大力发展冗余技术、容错技术，以及模块的热插拔功能，保障 PLC 能够长时间的可靠运行。

4）编程软件标准化。长期以来，PLC 的生产厂家各自为战，各产品在硬件结构和软件体系上都是封闭的，不对外开放，因而导致硬件互不通用、软件互不兼容，为用户带来很大的不便。为此，国际电工委员会（IEC）制定了 IEC 1131 标准以引导 PLC 向标准化方向发展。这个标准包含了 5 个部分，从 PLC 的定义等一般信息，到装备与测试、编程语言、用户规则、通信规范等，力图通过一系列的标准来规范各个厂家的产品。目前，有很多厂家都推出了符合 IEC 1131-3 标准的软件系统，例如西门子公司的 STEP 7 软件包就提供符合 IEC 1131-3 标准的指令集。

5）编程软件和语言向高层次发展。PLC 的编程语言在原有的梯形图、顺序功能图、指令表语言的基础上，不断丰富并向高层次发展。大部分厂商都提供可在个人计算机上运行的开发软件包，开发环境完备且友好，可向开发人员提供丰富的帮助信息以及调试、诊断、模拟仿真等功能。例如西门子公司的 STEP 7 软件包，运行在 Windows 环境下，在编程的过程中可随时查询指令，其内容与详细程度与编程手册相同。

小型化 PLC 的发展方向是体积减小、成本下降、功能齐全、性能提高、简单易用。其针对目标是取代广泛分布在企业和民用领域的小规模继电器系统，以及需要采用逻辑顺序控制的小规模场合。其特点是安装方便、可靠性高、开发和改造周期短。

二、PLC 的特点

PLC 的产生是基于工业控制的需要，是面向工业控制领域的专用设备，它具有以下几个特点：

1）可靠性高，抗干扰能力强。用程序来实现的逻辑顺序和时序，最大限度地取代传统继电器系统中的硬件线路，大量减少机械触点和连线的数量，单从这一角度而言，PLC 在可靠性上优于继电器系统是明显的。

在抗干扰性能方面，PLC 在结构设计、内部电路设计、系统程序执行等方面都给予了充分的考虑。例如对主要器件和部件用导磁良好的材料进行屏蔽、对供电系统和输入电路采用多种形式的滤波、I/O 回路与微处理器电路之间用光耦合器隔离、系统软件具有故障检测功能、信息保护和恢复、循环扫描时间的超时警戒等。

2）灵活性强，控制系统具有良好的柔性。当生产工艺和流程进行局部的调整和改动时，通常只需要对 PLC 的程序进行改动，或者配合以外围电路的局部调整即可实现对控制系统的改造。

3）编程简单，使用方便。梯形图语言是 PLC 的最重要也是最普及的一种编程语言，其电路符号和表达方式与继电器电路原理图相似，电气技术人员和技术工人可以很快地掌握梯形图语言，并用来编制用户程序。

4）控制系统易于实现，开发工作量少，周期短。由于 PLC 的系列化、模块化、标准化，以及良好的扩展性和联网性能，在大多数情况下，PLC 系统都是一个较好的选择，它不仅能够完成多数情况下的控制要求，还能够大量节省系统设计、安装、调试的时间和工作量。

5）维修方便。PLC 有完善的故障诊断功能，可以根据装置上的发光二极管和软件提供的故障信息，方便地查明故障源。由于 PLC 的体积小，并且有些是采用模块化结构，因而可以通过更换整机或模块迅速排除故障。

6）体积小，能耗低。由软件实现的逻辑控制，大量节省继电器、定时器，一台小型的 PLC 只相当于几个继电器的体积，控制系统所消耗的能量大大降低。

7）功能强，性能价格比高。用户程序实现的逻辑控制，所需要的继电器、中间继电器、定时器、计数器等功能元件，都由存储单元来替代，因而数量非常大，一台小型的 PLC 所具备的元件（软元件）数量就可达到成百上千个，相当于过去一个大规模甚至超大规模的继电器控制系统。另外，PLC 所提供的软元件的触点（例如软继电器）可以无限次使用，方便地实现复杂的控制功能。同时，PLC 的联网通信功能有利于实现分散控制、远程控制、集中管理等功能，与同等规模或成本的继电器控制系统相比，无论其功能和性能，都具有无可比拟的优势。

第二节　PLC 的组成

PLC 是微机技术和控制技术相结合的产物，是一种以微处理器为核心的用于控制的特殊计算

机，因此 PLC 的基本组成与一般的微机系统类似。

PLC 的硬件主要由中央处理器（Central Processing Unit，CPU）、存储器、输入单元、输出单元、通信接口、扩展接口、电源等部分组成。其中，CPU 是 PLC 的核心，输入单元与输出单元是连接现场输入/输出（I/O）设备与 CPU 之间的接口电路，通信接口用于与编程器、上位计算机等外设连接。图 1-1 是 PLC 的基本组成。

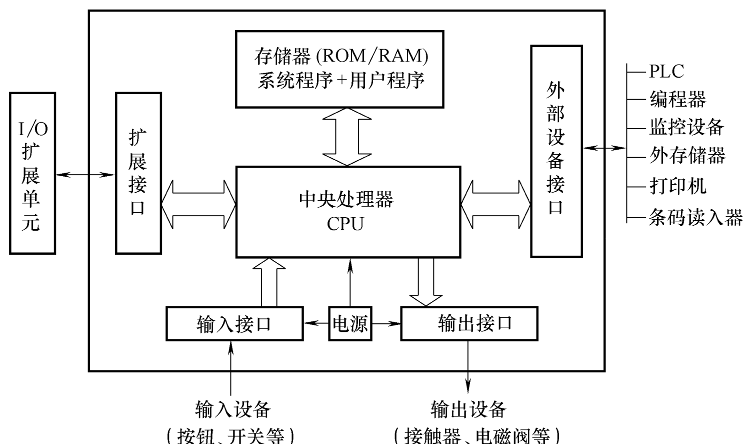


图 1-1 PLC 的基本组成

1. 中央处理单元（CPU）

同一般的微机一样，CPU 是 PLC 的核心。PLC 中所配置的 CPU 随机型不同而不同，常用的有三类：通用微处理器（如 Z80、8086、80286 等）、单片微处理器（如 8031、8096 等）和位片式微处理器（如 AMD29W 等）。小型 PLC 大多采用 8 位通用微处理器和单片微处理器；中型 PLC 大多采用 16 位通用微处理器或单片微处理器；大型 PLC 大多采用高速位片式微处理器。

目前，小型 PLC 为单 CPU 系统，而中、大型 PLC 则大多为双 CPU 系统，甚至有些 PLC 中 CPU 多达 8 个。对于双 CPU 系统，其中一个为字处理器，通常采用 8 位或 16 位处理器；另一个为位处理器，采用由各厂家设计制造的专用芯片。字处理器为主处理器，用于执行编程器接口功能，监视内部定时器，监视扫描时间，处理字节指令以及对系统总线和位处理器进行控制等。位处理器为从处理器，主要用于处理位操作指令和实现 PLC 编程语言向机器语言的转换。位处理器的采用，提高了 PLC 的速度，使 PLC 更好地满足实时控制要求。

在 PLC 中 CPU 按系统程序赋予的功能，指挥 PLC 有条不紊地进行工作，归纳起来主要有以下几个方面：

1）接收从编程器输入的用户程序和数据。

2）诊断电源、PLC 内部电路的工作故障和编程中的语法错误等。

3）通过输入接口接收现场的状态或数据，并存入输入映像寄存器或数据寄存器中。

4）从存储器逐条读取用户程序，经过解释后执行。

5）根据执行的结果，更新有关标志位的状态和输出映像寄存器的内容，通过输出单元实现输出控制。有些 PLC 还具有制表打印或数据通信等功能。

2. 存储器单元

存储器主要有两种：一种是可读/写操作的随机存储器（RAM），另一种是只读存储器（ROM、PROM、EPROM 和 EEPROM）。在 PLC 中，存储器主要用于存放系统程序、用户程序及

工作数据。

系统程序是由 PLC 的制造厂家编写的，与 PLC 的硬件组成有关，完成系统诊断、命令解释、功能子程序调用管理、逻辑运算、通信及各种参数设定等功能，提供 PLC 运行的平台。系统程序关系到 PLC 的性能，而且在 PLC 使用过程中不会变动，所以是由制造厂家直接固化在只读存储器 ROM、PROM 或 EPROM 中，用户不能访问和修改。

用户程序是随 PLC 的控制对象而定的，由用户根据对象生产工艺的控制要求而编制的应用程序。为了便于读出、检查和修改，用户程序一般存于 CMOS 静态 RAM 中，用锂电池作为后备电源，以保证掉电时不会丢失信息。为了防止干扰对 RAM 中程序的破坏，当用户程序经过调试，运行正常且不需要改变时，可将其固化在只读存储器 EPROM 中。现在有许多 PLC 直接采用 EEPROM 作为用户存储器。

工作数据是 PLC 运行过程中经常变化、经常存取的一些数据。存放在 RAM 中，以适应随机存取的要求。在 PLC 的工作数据存储器中，设有存放输入/输出继电器、辅助继电器、定时器、计数器等逻辑器件的存储区，这些器件的状态都是由用户程序的初始设置和运行情况而确定的。根据需要，部分数据在掉电时用后备电池维持其现有的状态，这部分在掉电时可保存数据的存储区域称为保持数据区。

由于系统程序及工作数据与用户无直接联系，所以在 PLC 产品样本或使用手册中所列存储器的形式及容量是指用户程序存储器。当 PLC 提供的用户存储器容量不够用时，许多 PLC 还提供有存储器扩展功能。

3. 电源单元

电源单元将外界提供的电源转换成 PLC 的工作电源后，提供给 PLC。有些电源单元也可以作为负载电源，通过 PLC 的 I/O 接口向负载提供直流 24V 电源。PLC 的电源一般采用开关电源，输入电压范围宽，抗干扰能力强。电源单元的输入与输出之间有可靠的隔离，以确保外界的扰动不会影响到 PLC 的正常工作。

电源单元还提供掉电保护电路和后备电池电源，以维持部分 RAM 存储器的内容在外界电源断电后不会丢失。在面板上通常有发光二极管指示电源的工作状态，便于判断电源工作是否正常。

4. 输入/输出单元

输入/输出单元通常也称 I/O 单元或 I/O 模块，是 PLC 与工业生产现场之间的连接部件。PLC 通过输入接口可以检测被控对象的各种数据，以这些数据作为 PLC 对被控制对象进行控制的依据；同时 PLC 又通过输出接口将处理结果送给被控制对象，以实现控制的目的。

由于外部输入设备和输出设备所需的信号电平是多种多样的，而 PLC 内部 CPU 处理的信息只能是标准电平，所以 I/O 接口要实现这种转换。I/O 接口一般都具有光电隔离和滤波功能，以提高 PLC 的抗干扰能力。另外，I/O 接口上通常还有状态指示，工作状况直观，便于维护。PLC 提供了多种操作电平和驱动能力的 I/O 接口，有各种各样功能的 I/O 接口供用户选用。I/O 接口的主要类型有：数字量（开关量）输入、数字量（开关量）输出、模拟量输入、模拟量输出等。

5. 接口单元

接口单元包括扩展接口、通信接口、编程器接口和存储器接口等。

PLC 的 I/O 单元也属于接口单元的范畴，它完成 PLC 与工业现场之间电信号的往来联系。除此之外，PLC 与其他外界设备和信号的联系都需要相应的接口单元。

（1）I/O 扩展接口

I/O 扩展接口用于扩展输入/输出点数，当主机的 I/O 通道数量不能满足系统要求时，需要

增加扩展单元，这时需要用到 I/O 扩展接口将扩展单元与主机连接起来。西门子公司 S7-300/400 中的接口模块（例如 IM365、IM360/361 等）就是专用于连接中央机架和扩展机架的扩展接口。

（2）通信接口

PLC 配有各种通信接口，这些通信接口一般都带有通信处理器。PLC 通过这些通信接口可与监视器、打印机、其他 PLC、计算机等设备实现通信。PLC 与打印机连接，可将过程信息、系统参数等输出打印；与监视器连接，可将控制过程图像显示出来；与其他 PLC 连接，可组成多机系统或连成网络，实现更大规模的控制；与计算机连接，可组成多级分布式控制系统，实现控制与管理相结合。另外，远程 I/O 系统也必须配备相应的通信接口模块。

（3）编程器接口

编程器接口是连接编程器的，PLC 本体通常是不带编程器的。为了能对 PLC 编程和监控，PLC 上专门设置有编程器接口。通过这个接口可以连接各种形式的编程装置，还可以利用此接口做通信、监控工作。

（4）存储器接口

存储器接口是为了扩展存储区而设置的。用于扩展用户程序存储区和用户数据参数存储区，可以根据使用的需要扩展存储器，其内部也是接到总线上的。

（5）智能接口模块

智能接口模块是一个独立的计算机系统，它有自己的 CPU、系统程序、存储器以及与 PLC 系统总线相连的接口。它作为 PLC 系统的一个模块，通过总线与 PLC 相连，进行数据交换，并在 PLC 的协调管理下独立地进行工作。

PLC 的智能接口模块种类很多，如：高速计数模块、闭环控制模块、运动控制模块、中断控制模块等。

6. 外部设备

PLC 的外部设备种类很多，总体来说可以概括为四大类：编程设备、监控设备、存储设备、输入/输出设备。

（1）编程设备

编程设备的作用是编辑、调试、输入用户程序，也可在线监控 PLC 内部状态和参数，与 PLC 进行人机对话。它是开发、应用、维护 PLC 不可缺少的工具。编程装置可以是专用编程器，也可以是配有专用编程软件包的通用计算机系统。专用编程器是由 PLC 厂家生产，专供该厂家生产的某些 PLC 产品使用，它主要由键盘、显示器和外存储器接插口等部件组成。专用编程器有简易编程器和智能编程器两类。

简易编程器只能联机编程，而且不能直接输入和编辑梯形图程序，需将梯形图程序转化为指令表程序才能输入。简易编程器体积小、价格便宜，它可以直接插在 PLC 的编程插座上，或者用专用电缆与 PLC 相连，以方便编程和调试。有些简易编程器带有存储盒，可用来储存用户程序，如三菱的 FX-20P-E 简易编程器。

智能编程器又称图形编程器，本质上它是一台专用便携式计算机，如三菱的 GP-80FX-E 智能型编程器。它既可联机编程，又可脱机编程。可直接输入和编辑梯形图程序，使用更加直观、方便，但价格较高，操作也比较复杂。大多数智能编程器带有磁盘驱动器，提供录音机接口和打印机接口。

专用编程器只能对指定厂家的几种 PLC 进行编程，使用范围有限，价格较高。同时，由于 PLC 产品不断更新换代，所以专用编程器的生命周期也十分有限。因此，现在的趋势是使用以个人计算机为基础的编程装置，用户只要购买 PLC 厂家提供的编程软件和相应的硬件接口装置。

这样，用户只用较少的投资即可得到高性能的 PLC 程序开发系统。

基于个人计算机的程序开发系统功能强大。它既可以编制、修改 PLC 的梯形图程序，又可以监视系统运行、打印文件、系统仿真等。配上相应的软件还可实现数据采集和分析等许多功能。

（2）监控设备

PLC 将现场数据实时上传给监控设备，监控设备则将这些数据动态实时显示出来，以便操作人员和技术人员随时掌握系统运行的情况，操作人员能够通过监控设备向 PLC 发送操控指令，也把具有这种功能的设备称为人机界面。PLC 厂家通常都提供专用的人机界面设备，目前使用较多的有操作屏和触摸屏等。这两种设备均采用液晶显示屏，通过专用的开发软件可设计用户工艺流程图，与 PLC 联机后能够实现现场数据的实时显示。操作屏同时还提供多个可定义功能的按键，而触摸屏则可以将控制键直接定义在流程图的画面中，使得控制操作更加直观。

（3）存储设备

存储设备用于保存用户数据，避免用户程序丢失。有存储卡、存储磁带、软磁盘或只读存储器等多种形式，配合这些存储载体，有相应的读写设备和接口部件。

（4）输入/输出设备

用于接收信号和输出信号的专用设备。例如条码读入器、打印机等。

第三节　PLC 的工作原理

PLC 是基于电子计算机的工业控制器，从 PLC 产生的背景来看，PLC 系统与继电器控制系统有着极深的渊源，因此一个继电器控制系统必然包含：输入部分、逻辑电路部分和输出部分。输入部分的组成元件大体上是各类按钮、转换开关、行程开关、接近开关、光电开关等；输出部分则是各种电磁阀线圈、接触器、信号指示灯等执行元件。将输入与输出联系起来的就是逻辑电路部分，一般由继电器、计数器、定时器等器件的触点、线圈按照要求的逻辑关系连接而成，能够根据一定的输入状态输出所要求的控制动作。

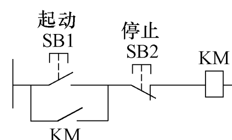


图 1-2　继电器起动/停止控制电路

PLC 系统也同样包含这三部分，唯一的区别是，PLC 的逻辑电路部分用软件来实现，用户所编制的控制程序体现了特定的输入/输出逻辑关系。举例来说，如图 1-2 所示为一个典型的起动/停止控制电路，由继电器元件组成。电路中有两个输入，分别为起动按钮（SB1）、停止按钮（SB2）；1 个输出为接触器 KM。图中的输入/输出逻辑关系由硬件连线实现。

当用 PLC 来完成这个控制任务时，可将输入条件接入 PLC，而用 PLC 的输出单元驱动接触器 KM，它们之间要满足的逻辑关系由程序实现。与图 1-2 等效的 PLC 等效电路如图 1-3 所示。

两个输入按钮信号经过 PLC 的接线端子进入输入接口电路，PLC 的输出经过输出接口、输出端子驱动接触器 KM；用户程序所采用的编程语言为梯形图语言。两个输入分别接入 X403 和 X407 端口，输出所用端口为 Y432，图中只画出 8 个输入端口和 8 个输出端口，实际使用时可任意选用。输入映像对应的是 PLC 内部的数据存储器，而非实际的继电器线圈。

图中的 X400 ~ X407、Y430 ~ Y437 分别表示输入、输出端口的地址，也对应着存储器空间中特定的存储位，这些位的状态（ON 或者 OFF）表示相应输入、输出端口的状态。每一个输入、输出端口的地址是唯一固定的，PLC 的接线端子号与这些地址一一对应。由于所有的输入、输出状态都是由存储器位来表示的，它们并不是物理上实际存在的继电器线圈，所以常称它们为

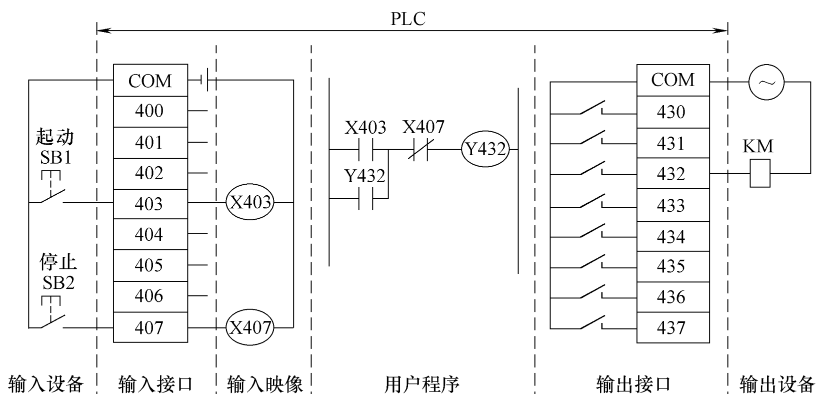


图 1-3　PLC 等效电路

“软元件”，它们的常开、常闭触点可以在程序中无限次使用。

PLC 的工作过程以循环扫描的方式进行，当 PLC 处于运行状态时，它的运行周期可以划分为 3 个基本阶段：输入采样阶段、程序执行阶段、输出刷新阶段。

1. 输入采样阶段

在这个阶段，PLC 逐个扫描每个输入端口，将所有输入设备的当前状态保存到相应的存储区，我们把专用于存储输入设备状态的存储区称为输入映像寄存器，图 1-3 中以线圈形式标出的 X403、X407，实际上是输入映像寄存器的形象比喻。

输入映像寄存器的状态被刷新后，将一直保存，直至下一个循环才会被重新刷新，所以当输入采样阶段结束后，如果输入设备的状态发生变化，也只能在下一个周期才能被 PLC 接收到。

2. 程序执行阶段

PLC 将所有的输入状态采集完毕后，进入用户程序的执行阶段。所谓用户程序的执行，并非是系统将 CPU 的工作交由用户程序来管理，CPU 所执行的指令仍然是系统程序中的指令。在系统程序的指示下，CPU 从用户程序存储区逐条读取用户指令，经解释后执行相应动作，产生相应结果，刷新相应的输出映像寄存器，期间需要用到输入映像寄存器、输出映像寄存器的相应状态。

当 CPU 在系统程序的管理下扫描用户程序时，按照先上后下、先左后右的顺序依次读取梯形图中的指令。以图 1-3 中的用户程序为例，CPU 首先读到的是常开触点 X403，然后在输入映像寄存器中找到 X403 的当前状态，接着从输出映像寄存器中得到 Y432 的当前状态，两者的当前状态进行“或”逻辑运算，结果暂存；CPU 读到的下一条梯形图指令是 X407 的常闭触点，同样从输入映像寄存器中得到 X407 的状态，将 X407 常闭触点的当前状态与上一步的暂存结果进行逻辑“与”运算，最后根据运算结果得到输出线圈 Y432 的状态（“ON”或者“OFF”），并将其保存到输出映像寄存器中，也就是对输出映像寄存器进行了刷新。请注意，在程序执行过程中用到了 Y432 的状态，这个状态是上一个周期执行的结果。

当用户程序被完全扫描一遍后，所有的输出映像都被依次刷新，系统将进入下一个阶段，即输出刷新阶段。

3. 输出刷新阶段

在这个阶段，系统程序将输出映像寄存器中的内容传送到输出锁存器中，经过输出接口或输出端子输出，驱动外部负载。输出锁存器一直将状态保持到下一个循环周期，而输出映像寄存器

的状态在程序执行阶段是动态的。

4. 总结

根据上述过程的描述，可对 PLC 工作过程的特点总结如下：

1）PLC 采用集中采样、集中输出的工作方式，这种方式减少了外界干扰的影响。

2）PLC 的工作过程是循环扫描的过程，循环扫描时间的长短取决于指令执行速度、用户程序的长度等因素。

3）输出对输入的响应有滞后现象。PLC 采用集中采样、集中输出的工作方式，当采样阶段结束后，输入状态的变化将要等到下一个采样周期才能被接收，因此这个滞后时间的长短又主要取决于循环周期的长短。此外，影响滞后时间的因素还有输入电路滤波时间、输出电路的滞后时间等。

4）输出映像寄存器的内容取决于用户程序扫描执行的结果。

5）输出锁存器的内容，由上一次输出刷新期间输出映像寄存器中的数据决定。

6）PLC 当前实际的输出状态，由输出锁存器的内容决定。

需要补充说明的是，当系统规模较大、I/O 点数众多、用户程序比较长时，单纯采用上面的循环扫描工作方式会使系统的响应速度明显降低，甚至会丢失、错漏高频输入信号，因此大多数大中型 PLC 在尽量提高程序指令执行速度的同时，也采取了一些其他措施来加快系统响应速度。例如采用定周期输入采样、输出刷新，直接输入采样、直接输出刷新，中断输入、输出，或者开发智能 I/O 模块，模块本身带有 CPU，可以与主机的 CPU 并行工作，分担一部分任务，从而加快整个系统的执行速度。

第四节　PLC 的硬件基础

I/O 单元是组成 PLC 系统的重要环节，本节以介绍 I/O 单元的硬件电路为主，在此基础上简单介绍 PLC 系统的硬件配置。应当说明的是，不同 PLC 在硬件的具体实现方案上总是有区别的，本节的任务是讨论一般性的原理，而非某一具体型号的结构特征，本书后续章节将针对不同型号的 PLC，分别介绍其特点。

一、PLC 的 I/O 模块

正如本章第二节中所介绍的，PLC 的输入/输出部分，可以分为数字量 I/O（DI/DO）和模拟量 I/O（AI/AO）两大类。

1. 数字量 I/O（DI/DO）

PLC 一般总是将输入、输出分成若干组，每组共用一个输入、输出端口，下面分别介绍数字量输入、输出电路的具体形式。

（1）数字量输入单元

数字量输入电路有多种形式，能分别适用于直流和交流的数字输入量。而在直流数字量的输入电路中，根据具体的电路形式又有源型和漏型之别。图 1-4 是漏型数字量输入电路示意图。

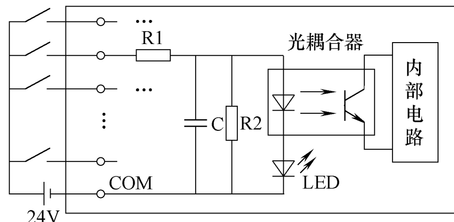


图 1-4　漏型数字量输入电路示意图

在图 1-4 中，若干个输入点组成一组，共用一个公共端 COM。每一个点都构成一个回路，图中只画出了一路。回路的电流流向是从输入端口流入 PLC，从公共端流出。图中的电阻 R2 和

电容 C 构成 RC 滤波电路，光耦合器将现场信号与 PLC 内部电路隔离，并且将现场信号的电平（图中为 DC24V）转换为 PLC 内部电路可以接受的电平。发光二极管（LED）用来指示当前数字量输入信号的高、低电平状态。

源型输入电路的形式与图 1-4 基本相似，不同之处在于光耦合器、发光二极管、DC24V 电源均反向，电流流向是从公共端 COM 流入 PLC，从信号端流出。

目前，有很多 PLC 采用双向光耦合器，并且使用两个反向并联的发光二极管，这样一来，DC24V 电源的极性可以任意接，电流的流向也可以是任意的。

交流数字量输入电路也有多种形式，有些采用桥式整流电路将交流信号转换成直流，然后经过光耦合器隔离输入内部电路；而有些 PLC 则直接使用双向光耦合器和双向发光二极管，从而省去了桥式整流电路。图 1-5 是带整流桥的输入电路示意图。

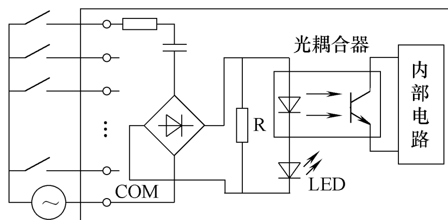


图 1-5　带整流桥的输入电路示意图

（2）数字量输出单元

PLC 的数字量输出有三种形式：继电器模式、晶体管模式、晶闸管模式，分别用于驱动不同形式的负载。图 1-6 给出了继电器输出模式的原理图，图中的 KA 为输出继电器，继电器输出模式可以带交流、直流两种负载。

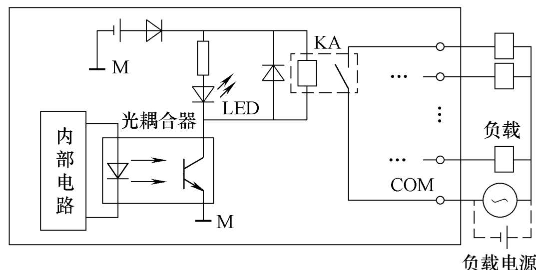


图 1-6　继电器输出电路示意图

2. 模拟量 I/O（AI/AO）

PLC 的模拟量 I/O 接口用于处理连续变化的电压或电流信号，在过程控制领域以及数据采集及监控系统中用途极广。

（1）模拟量输入单元

传感器将被控对象中连续变化的物理量（例如温度、压力、流量、速度等）转换成对应的连续电量（电压或电流）并送给 PLC，PLC 的模拟量输入单元将其转换成数字量后，CPU 可对其进行运算处理。因此，模拟量输入单元的核心部件是 A-D 转换器，对于多路输入的模块，需要多路开关配合使用。图 1-7 为具有 8 个输入通道的模拟量输入单元原理框图。

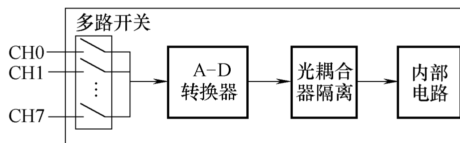


图 1-7　8 通道模拟量输入单元原理框图

模拟量输入信号可以是电压或电流，在选型时要考虑输入信号的范围以及系统要求的 A-D 转换精度。常见的输入范围有 DC ±10V、0～10V、±20mA、4～20mA 等，转换精度有 8 位、10 位、11 位、12 位、16 位等，PLC 生产厂家的相关技术手册都会提供这些参数。此外，选型时还需要考虑接线形式是否与传感器匹配。

（2）模拟量输出单元

模拟量输出的过程与输入正相反，它将 PLC 运算处理过的二进制数字转换成相应的电量（例如 4～20mA、0～10V 等），输出至现场的执行机构，它的核心部件是 D-A 转换器。图 1-8 为

模拟量输出单元的原理框图。

模拟量输出单元的主要技术指标同样包括输出信号形式（电压或电流）、输出信号范围（例如4～20mA、0～10V等），以及接线形式等，在选型时要充分考虑到这些因素与工业现场执行元件相互结合的问题。

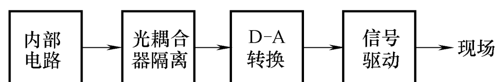


图1-8　模拟量输出单元原理框图

二、PLC的配置

PLC的品种繁多，其结构形式、性能、容量、指令系统、编程方法等各有特点，适用场合也各有侧重。站在硬件选型的角度，首先需要考虑的是设备容量与性能是否与任务相适应；其次要看PLC运行速度是否能够满足实时控制的要求。

所谓设备容量，主要是指系统I/O点数的多少以及扩充的能力。对于纯开关量控制的应用系统，如果对控制速度的要求不高，比如单台机械的自动控制，可选用小型一体化PLC，例如三菱公司的FX2N系列PLC。

对于以开关量控制为主，带有部分模拟量控制的应用系统，如工业中常遇到的温度、压力、流量、液位等，应配备模拟量I/O（AI/AO），并且选择运算功能较强的小型PLC，例如西门子公司的S7-200系列PLC。

对于比较复杂，控制功能要求较高的系统，比如需要PID调节、位置控制、高速计数、通信联网等功能时，应当选用中、大型PLC，这一类PLC多为模块式结构，除了基本的模块外，还提供专用的特殊功能模块。当系统的各个部分分布在不同的地域时，可以利用远程I/O组成分布式控制系统。适合这一类型的产品有西门子公司的S7-300/400系列PLC等。

PLC的输出控制相对于输入的变化总是有滞后的，最大可至2～3个循环周期，这对于一般的工业控制是允许的。但有些系统的实时性要求较高，不允许有较大的滞后时间，在这种要求比较高的场合，必须格外重视PLC的指令执行速度指标，选择高性能、模块式结构的PLC较为理想。例如西门子公司的S7-300/400 PLC，浮点运算指令的执行时间可以达到微秒级，另一个好处是可以配备专用的智能模块，这些模块都自带CPU独立完成操作，可大大提高控制系统的实时性。

一体化机型的PLC将电源部件集成在主机内，只需从电网引入外界电源即可，扩展单元的用电可通过扩展电缆馈送。模块式PLC通常需要专用的电源模块，在选择电源模块时要考虑功率的问题，可以通过查阅模块技术手册得到各个模块的功耗，其总和再加上裕量就是选择电源模块的依据。注意，有些情况下需要PLC电源通过I/O单元驱动传感器和负载，这一部分功耗也必须考虑在内。

第五节　PLC的软件基础

PLC是一种通用的、商业化的工业控制计算机，与个人计算机相仿，用户程序必须在系统程序的管理下才能运行。本节首先介绍PLC系统监控程序的运行情况，然后再介绍用户指令系统的相关内容。

一、系统监控程序

系统监控程序的运行从设备上电开始，经过初始化程序后进入循环执行阶段。在循环执行阶段要完成的操作有四大类：以故障诊断、通信处理为主的公共操作；联系工业现场的数据输入、输出操作；执行用户程序的操作；服务于外部设备的操作。图1-9是系统监控程序执行过程框

图，图中的输入刷新、用户程序执行、输出刷新三部分内容在第三节专门讲过，这里只介绍其他几部分。

1. 初始化程序

作用是清零各个标志寄存器，清零输入、输出映像寄存器，清零所有计数器，复位定时器等，即为 PLC 开始正常工作“清理现场”。

2. CPU 自诊断

自诊断主要包括检查电源电压是否正常，I/O 单元的连接是否正常，用户程序是否存在语法错误，对监控定时器定期复位等。监控定时器又常被称为“看门狗”（Watch Dog Timer，WDT），其定时时间略长于整个程序的循环周期，系统程序总在某一固定阶段对它重新装入定时初值，所以只要系统工作正常，监控定时器就永远不会申请定时到中断。否则，如果监控定时器申请定时时间到中断，就一定意味着系统的某处出现了问题，系统会响应其中断，并在中断处理程序中对故障信息做相应处理。

3. 通信信息处理

这个阶段 PLC 要完成与网络及总线上其他设备的通信任务，包括与 PLC、计算机、智能 I/O 模块、数字处理器（Data Processing Unit，DPU）等设备之间的信息交换。

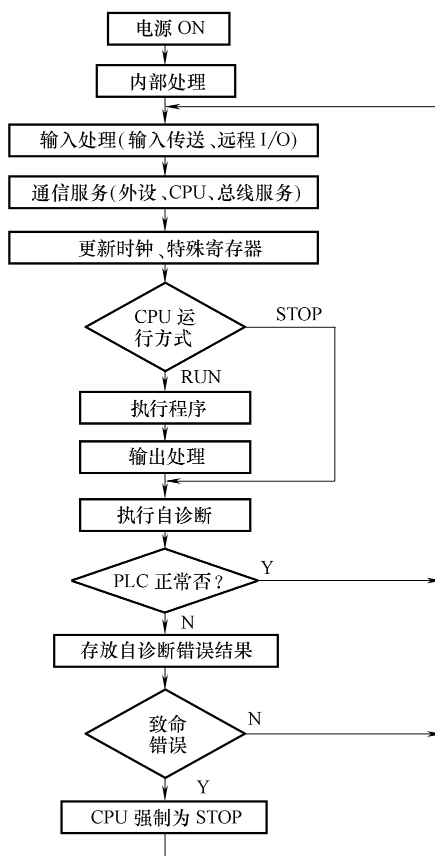


图 1-9 系统监控程序执行过程框图

4. 外部设备服务

PLC 在这个阶段与外部设备交换信息，包括编程器、图形监视器（监控设备）、打印机等。PLC 允许在线编程，能够与人机界面实时交换信息，所以要在每个循环周期内执行此项操作。

二、用户应用程序

用户程序是由用户编写的，能够完成系统控制任务的指令序列。不同厂家的 PLC 会提供不同的指令集，但基本的编程元件和编程形式有许多共同之处。

1. PLC 的编程元件

PLC 的编程元件也称为逻辑部件，是 PLC 指令系统中的基本要素，PLC 指令系统通常都提供以下逻辑部件：

（1）继电器

输入、输出映像寄存器里的每一位，在指令系统中都对应一个固定的编号，在图形编程语言（例如梯形图语言）中形象地用继电器线圈来表示，因此也常称之为输入继电器、输出继电器。同时为了满足对复杂逻辑关系的编程要求，还提供大量的中间辅助继电器，它们也对应存储器中的某一固定区域。这些继电器都是所谓的“软元件”，它们的状态用一个二进制位就可以表示，1 对应“ON”状态，0 对应“OFF”状态。

（2）定时器

类似于继电器逻辑电路中的时间继电器，有延时接通、延时断开、脉冲定时等多种形式，可以组成复杂的时间顺序逻辑。定时器指令一般由线圈、定时时间设定值和当前计时值组成，PLC专门在存储器中开辟出一个区域，用以保存各个定时器线圈当前的状态（“ON”或“OFF”）以及时间的设定值和当前值。定时器的常开、常闭触点可以在用户程序中无限次使用。

（3）计数器

用软件实现的计数器指令，用于实现脉冲计数功能，有递减计数、递增计数等形式，不同的PLC在计数器数量、计数长度等方面都有所区别。计数器指令一般包含计数器线圈、计数值设定、计数器复位、计数信号输入、当前计数值等。计数器的常开、常闭触点可以在用户程序中无限次使用。

（4）触发器

该指令用于对状态位的置1和清零，状态位即为触发器线圈，它的“ON”状态一旦触发可以自保持，直至复位条件满足才变为“OFF”状态。触发器的常开、常闭触点可以无限次使用。

（5）其他元器件及指令

除上述四种逻辑元件之外，PLC指令系统一般还提供移位寄存器、数据寄存器、边沿检测、比较、运算、ASCII码处理以及数制转换等多种指令。

2. PLC的编程语言

常用的编程语言有梯形图语言、语句表语言、功能块图等。

（1）梯形图语言（LAD）

这是一种使用最广泛的语言，与继电器电路图非常相似，具有直观易懂的优点。前面介绍的编程元件以及它们的线圈语言、触点等，都是基于梯形图语言而言的。下面通过一个简单的继电器逻辑电路和与之对应的梯形图语言程序的对比，来说明梯形图语言的应用，图1-10中，图1-10a为继电器逻辑电路，图1-10b为梯形图语言程序。

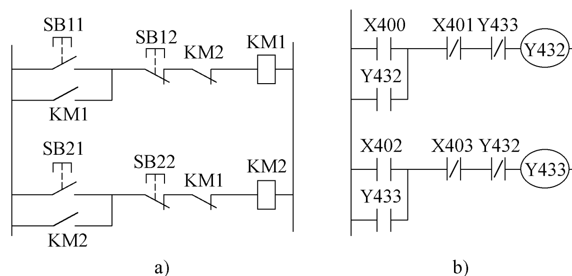


图1-10　梯形图语言

a）继电器逻辑电路　b）梯形图语言程序

图1-10a中的按钮SB11、SB12、SB21、SB22分别接入PLC的数字量输入点X400、X401、X402、X403，在实际接法上均使用按钮的常开触点；接触器线圈KM1、KM2分别由PLC的数字量输出点Y432、Y433驱动。

由图1-10可见，梯形图的形式与继电器电路图的形式很接近，其逻辑关系也是自上而下、自左而右展开的，左右两条竖线也称为母线。从左母线开始，按照控制要求依次连接各个触点，最后以输出线圈结束，称为一个逻辑行，或一个“梯级”，完整的用户程序就是由若干逻辑行构成的。

在阅读梯形图程序时，可按照继电器电路图纸的阅读习惯，对每一逻辑行来说，假设能量的流动由左母线向右流动，如果各触点的逻辑状态使得“能流”可以达到最右边的线圈，则该线圈的输出状态为“ON”，否则为“OFF”。

在编写梯形图程序时，有一些原则是被普遍遵守的，它们也都是出自继电器电路的设计原则，例如在一个逻辑行中不应串联两个线圈，同一个线圈不应出现在不同逻辑行中等。

（2）语句表语言（STL）

语句表语言（STL）类似于微机中汇编语言的助记符语言，由多条语句组成一个程序段，适合于经验丰富的程序员使用，可以实现某些用梯形图难以实现的功能。在使用简易编程器编程时，常常需要将梯形图转换为语句表才能输入 PLC。

（3）功能块图（FBD）

利用功能块图（Function Block Diagram）编程语言可以查看到像普通逻辑门图形的逻辑盒指令。它没有梯形图编程器中的触点和线圈，但有与之等价的指令，这些指令是作为盒指令出现的，程序逻辑由这些盒指令之间的连接决定。也就是说，一个指令（例如 AND 盒）的输出可以用来允许另一条指令（例如定时器），这样可以建立所需要的控制逻辑。这样的连接思想可以解决范围广泛的逻辑问题。FBD 编程语言有利于程序流的跟踪，但在目前使用较少。

（4）顺序功能流程图（SFC）

顺序功能流程图（Sequence Function Chart）编程是一种图形化的编程方法，亦称功能图。使用它可以对具有并发、选择等复杂结构的系统进行编程，许多 PLC 都提供了用于 SFC 编程的指令。目前，国际电工委员会（IEC ）也正在实施并发展这种语言的编程标准。

3. PLC 的程序结构

控制一个任务或过程，是通过在 RUN 方式下，使主机循环扫描并连续执行用户程序来实现的，用户程序决定了一个控制系统的功能。程序的编制可以使用编程软件在计算机或其他专用编程设备中进行（如图形输入设备），也可使用手编器。广义上的 PLC 程序由三部分构成：即用户程序、数据块和参数块。

（1）用户程序

用户程序是必选项。用户程序在存储器空间中也称为组织块，它处于最高层次，可以管理其他块，它是用各种语言（如 STL 、LAD 或 FBD 等）编写的用户程序。不同机型的 CPU，其程序空间容量也不同。用户程序的结构比较简单，一个完整的用户控制程序应当包含一个主程序、若干子程序和若干中断程序三部分。不同编程设备，对各程序块的安排方法也不同。

（2）数据块

数据块为可选部分，它主要存放控制程序运行所需的数据。

（3）参数块

参数块也是可选部分，它存放的是 CPU 组态数据，如果在编程软件或其他编程工具上未进行 CPU 的组态，则系统以默认值进行自动配置。

第六节　PLC 的性能指标及分类

PLC 产品种类繁多，其规格和性能也各不相同。对 PLC 的分类，通常根据其结构形式的不同、功能的差异和 I/O 点数的多少等进行大致分类。

一、按结构形式分类

根据 PLC 的结构形式，可将 PLC 分为整体式和模块式两类。

（1）整体式 PLC

整体式 PLC 是将电源、CPU、I/O 接口等部件都集中装在一个机箱内，具有结构紧凑、体积小、价格低的特点。小型 PLC 一般采用这种整体式结构。整体式 PLC 由不同 I/O 点数的基本单元（又称主机）和扩展单元组成。基本单元内有 CPU、I/O 接口、与 I/O 扩展单元相连的扩展口，以及与编程器或 EPROM 写入器相连的接口等。扩展单元内只有 I/O 和电源等，没有 CPU。基本单元和扩展单元之间一般用扁平电缆连接。整体式 PLC 一般还可配备特殊功能单元，如模

拟量单元、位置控制单元等，使其功能得以扩展。

（2）模块式 PLC

模块式 PLC 是将 PLC 各组成部分，分别做成若干个单独的模块，如 CPU 模块、I/O 模块、电源模块（有的含在 CPU 模块中）以及各种功能模块。模块式 PLC 由框架或基板和各种模块组成，模块装在框架或基板的插座上。这种模块式 PLC 的特点是配置灵活，可根据需要选配不同规模的系统，而且装配方便，便于扩展和维修。大、中型 PLC 一般采用模块式结构。

还有一些 PLC 将整体式和模块式的特点结合起来，构成所谓叠装式 PLC。叠装式 PLC 其 CPU、电源、I/O 接口等也是各自独立的模块，但它们之间是靠电缆进行连接，并且各模块可以一层层地叠装。这样，不但系统可以灵活配置，还可做得体积小巧。

二、按功能分类

根据 PLC 所具有的功能不同，可将 PLC 分为低档、中档、高档三类。

（1）低档 PLC

具有逻辑运算、定时、计数、移位以及自诊断、监控等基本功能，还可以有少量模拟量输入/输出、算术运算、数据传送和比较、通信等功能。主要用于逻辑控制、顺序控制或少量模拟量控制的单机控制系统。

（2）中档 PLC

除具有低档 PLC 的功能外，还具有较强的模拟量输入/输出、算术运算、数据传送和比较、数制转换、远程 I/O、子程序、通信联网等功能。有些还可增设中断控制、PID 控制等功能，适用于复杂控制系统。

（3）高档 PLC

除具有中档机的功能外，还增加了带符号算术运算、矩阵运算、位逻辑运算、平方根运算及其他特殊功能函数的运算、制表及表格传送功能等。高档 PLC 具有更强的通信联网功能，可用于大规模过程控制或构成分布式网络控制系统，实现工厂自动化。

三、按 I/O 点数分类

根据 PLC 的 I/O 点数的多少，可将 PLC 分为小型、中型和大型三类。

（1）小型 PLC

I/O 点数在 256 点以下的为小型 PLC。其中，I/O 点数小于 64 点的为超小型或微型 PLC。

（2）中型 PLC

I/O 点数在 256 点以上 2048 点以下的为中型 PLC。

（3）大型 PLC

I/O 点数为 2048 点以上的为大型 PLC。其中，I/O 点数超过 8192 点的为超大型 PLC。

在实际应用中，一般 PLC 功能的强弱与其 I/O 点数的多少是相互关联的，即 PLC 的功能越强，其可配置的 I/O 点数越多。因此，通常我们所说的小型、中型、大型 PLC，除指其 I/O 点数不同外，同时也表示其对应功能为低档、中档、高档。

第二章　西门子公司常用系统简介

本章将对西门子公司常用的重要系统做简单的介绍，主要有：SIMATIC S7-200/300/400 工业通信、HMI（人机界面）和 SIMATIC 工业软件。

第一节　SIMATIC PLC 控制器

西门子公司的 PLC 产品有 SIMATIC S7、M7 和 C7 等几大系列。

S7 系列是传统意义的 PLC 产品，其中的 S7-200 系列属于整体式小型 PLC，用于代替继电器的简单场合，也可以用于复杂的自动控制系统。S7-300 系列是模块化的中小型 PLC，最多可以扩展 32 个模块，适用于中等性能的控制要求。S7-400 是具有中高性能的 PLC，采用模块化无风扇设计，可以扩展 200 多个模块，适用于对可靠性要求极高的大型复杂控制系统。S7-300/400 可以组成 MPI（多点接口）、PROFIBUS 网络和工业以太网等。

SIMATIC M7-300/400PLC 采用与 S7-300/400 相同的结构，它可以作为 CPU 或功能模块使用。其显著特点是具有 AT 兼容计算机的功能，使用 S7-300/400 的编程软件 STEP7 和可选的 M7 软包，可以用 C、C + + 或 CFC（Continuous Function Chart，连续功能图）这类高级语言来对 M7-300/400PLC 编程。M7 适合于需要处理的数据量大，对数据管理、显示和实时性有较高要求的系统使用。

SIMATIC C7 由 S7-300PLC、HMI（人机界面）操作面板、I/O、通信和过程监控系统组成，整个控制系统结构紧凑，面向用户的配置、数据管理与通信集成在一起，具有很高的性能价格比。由于高度集成，节约了 30% 的安装空间，可以和谐地集成到 SIMATIC 控制产品家族中，保证正确的数据交换。

一、SIMATIC S7-200

SIMATIC S7-200 如图 2-1 所示。

SIMATIC S7-200 是一种低端 CPU。该 CPU 适用于机器与系统中的开环和闭环控制任务。它具有实时功能，并通过 PROFIBUS 或 PC/PPI 电缆以及一个自由可编程接口协议提供广泛的通信功能。SIMATIC S7-200 具有模块化扩展和集成 PID 闭环控制功能。使用编程软件 STEP 7 Micro/Win，可快速地进行编程和组态。

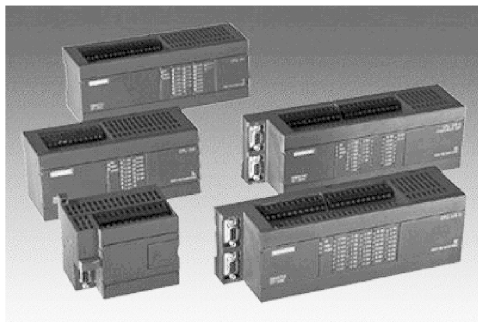


图 2-1　SIMATIC S7-200

二、SIMATIC S7-300

SIMATIC S7-300 如图 2-2 所示。

SIMATIC S7-300 系列 CPU 适用于低中端自动化解决方案。SIMATIC S7-300 的特性如下：

1）程序存储器可存储高达 85K 条指令。

2）高达 1024 点 I/O。

3）配有多点接口，可用于小型网络的配置，以及使用 PC/编程器进行组态。

4）执行时间快；CPU 可在 0.1ms 内执行 1024 条二进制指令。

5）通过带有集成背板总线的接口模块，可实现模块化配置和快速功能增。

6）丰富的数字量、模拟量、仿真模块以及通信功能模块和各种其他模块，可实现模块化扩展。

7）集成功能：CPU 312/314 IFM 集成有计数、定位、闭环控制和频率测量功能。

8）300 2-DP 系列还集成有 PROFIBUS 接口；CPU 也可用作从站。

9）丰富的数学公式处理。

10）在 CPU 操作系统中集成有循环 HMI 服务。

11）使用 STEP 7，可实现快速、简便的组态和编程。

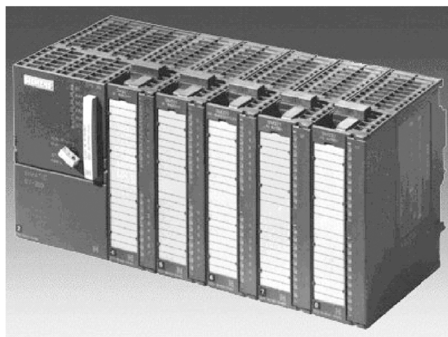


图 2-2　SIMATIC S7-300

12）使用 STEP 7，可实现丰富的诊断功能。带时间戳记的错误消息缓存器以及诊断模块，有助于用户查找故障。

三、SIMATIC S7-400

SIMATIC S7-400 如图 2-3 所示。

SIMATIC S7-400 系列 CPU 适用于中高端自动化解决方案（例如汽车、机床或仪表及控制系统）。SIMATIC S7-400 的特性如下：

1）程序存储器可存储高达 660K 条指令。

2）高达 131056 点数字量 I/O。

3）配有多点接口，可用于小型网络的配置，以及使用 PC/编程器进行组态。

4）执行时间快；CPU 可在 0.1μs 内执行 1024 条二进制指令。

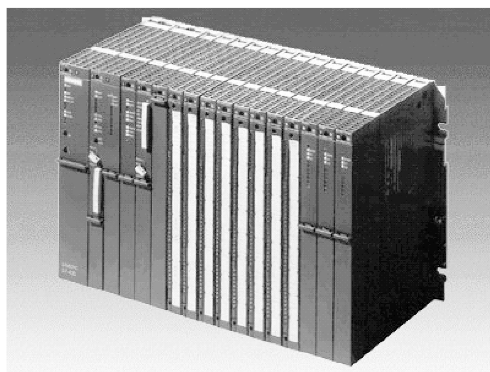


图 2-3　SIMATIC S7-400

5）通过带有集成背板总线的接口模块，可实现模块化配置和快速功能增强。

6）丰富的数字量、模拟量、仿真模块以及通信功能模块和各种其他模块。

7）S7-400 2-DP 系列还集成有 PROFIBUS 接口；采用 PROFIBUS-DP/FMS/PA 标准和工业以太网。

8）丰富的数学公式处理。

9）在 CPU 操作系统中集成有循环 HMI 服务。

10）使用 STEP 7，可实现快速、简便的组态和编程。

11）使用 STEP 7，可实现丰富的诊断功能。带时间戳记的错误消息缓存器以及诊断模块，有助于用户查找故障。

第二节　工 业 通 信

通过 SIMATIC 网络 PROFIBUS、以太网或 TCP/IP，可实现从过程控制系统到现场级的通信。SIMATIC NET 系列提供各种性能等级的产品供用户选用。配有可操作性系统接口，可实现各层级

的数据交换，包括各种自动化站和各种设备。

网络金字塔如图 2-4 所示。

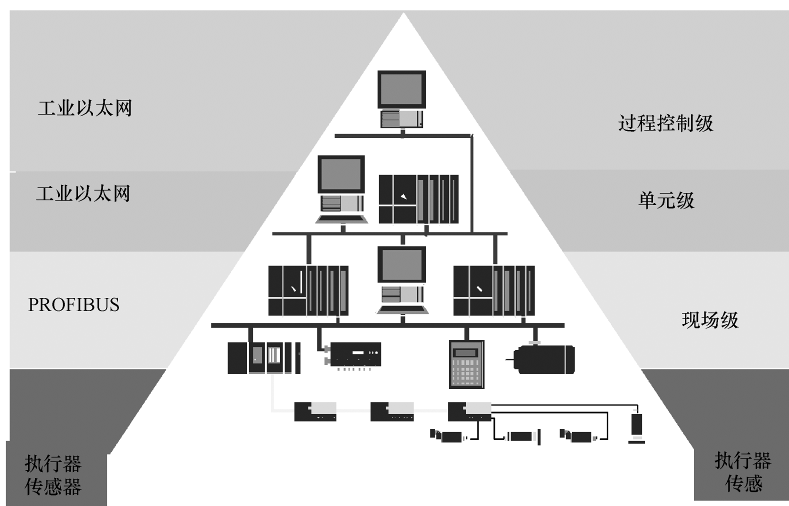


图 2-4 网络金字塔

一、工业以太网

通过工业以太网，可实现过程控制级的通信。传输速率为 10～100Mbit/s，通过快速以太网，可实现关联节点之间的快速数据交换。通信传输可通过同轴电缆、双绞线或光纤来实现。以太网的优点如下：通过简单地连接就能进行快速的调试；运行期间也能对现有系统进行扩展，可用性高；采用传输速率可调的交换技术，系统性能可根据规模伸缩；可连接不同的网络应用领域，如加工应用和办公应用；通过与 WAN（广域网），类似于 ISDN 或因特网，可实现公司范围内的连接；通过持续的兼容性开发，实现投资安全。

工业以太网部件如图 2-5 所示。

二、现场总线 PROFIBUS

PROFIBUS 用于将现场设备（例如分布式 I/O 或驱动装置）连接到诸如 SIMATIC S7 等的自动化系统。PROFIBUS 是一种高性能的开放式现场总线系统，响应时间快，配有开放式接口，可用于不同的协议网络。通过 PROFIBUS-DP，可采用电气或光学传输连接分布式 I/O，传输速率

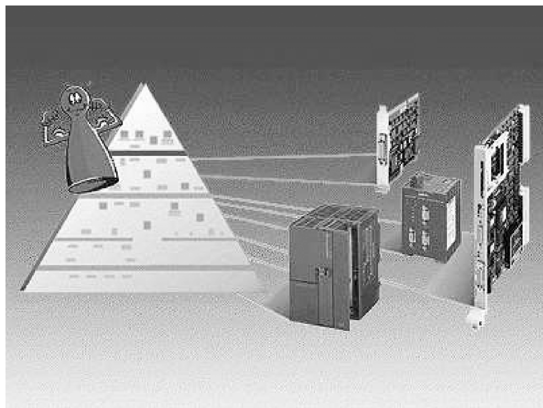


图 2-5 工业以太网部件

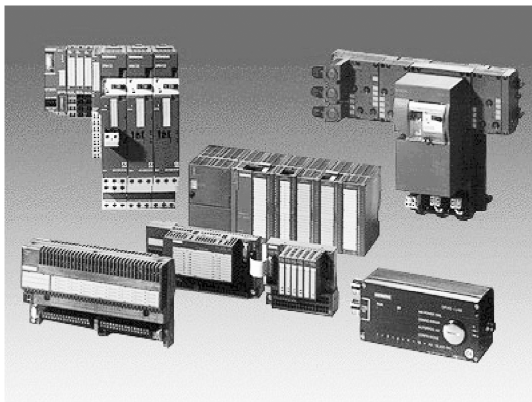


图 2-6 工业以太网组件

高达 12 Mbit/s。PROFIBUS-PA 是一种本安型 PROFIBUS，用于有爆炸性危险的应用场合（例如化工工业）。PROFIBUS-FMS 不仅可以用于现场级的上位系统级，还可用于对实时性要求不高的单元级和/或过程控制级。提供有丰富的各种协议网络产品，可连接到 PROFIBUS。其组态和编程仍使用软件 STEP 7，具有各种诊断功能。

工业以太网组件如图 2-6 所示。

三、AS-i 电缆连接

在现场级存在着各种不同的部件：阀门、执行器和驱动器。这些部件必须依靠自动化技术进行控制。为此，目前可采用 AS-Interface 分布式 I/O。执行器和传感器通过一根双芯电缆与控制器连接。数据和电源也通过该电缆进行传输。AS-Interface 的优点：无面板设计；防护等级高，IP65；直接从某个位置进行接线；可使用二线电缆实现更加简单和灵活的结构，无需具备专门知识；通过 DP/AS 接口进行 PROFIBUS 连接。

AS-接口如图 2-7 所示。

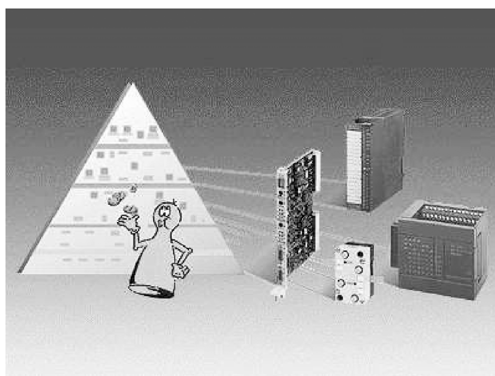


图 2-7　AS-接口

第三节　人 机 界 面

SIMATIC HMI（人机界面）组件可用作机器和用户之间的接口。各种功能、开关或过程值都可显示在操作员面板或触摸面板上。通过这种可视化，可很容易地显示错误消息或测量值。过程的光学检测功能减轻了用户的操作，并能快速地知道其操作的效果。

人机界面分为四类：

1）按钮面板（PP 7 和 PP 17），用于常规操作现场的创新性替代产品。

2）文本显示器（OP 3、OP 7 和 OP 17），用于机器中的操作和监控。

3）图形显示器（OP 25、OP 37、TP 27 和 TP 37），可使机器中操作和监控更加舒适。

4）基于 Windows 系统，例如 OP 37 pro、MP 370 或 MP 270，可用于机器中的操作。

这些装置都可使用组态工具 ProTool 进行组态。根据装置类型，该组态工具提供有三种不同性能的版本。这种人机界面可通过 MPI 或 PROFIBUS-DP 直接连接到自动化系统。借助于组态好的功能开关，命令按钮或显示元件都可直接访问 CPU。

人机界面如图 2-8 所示。

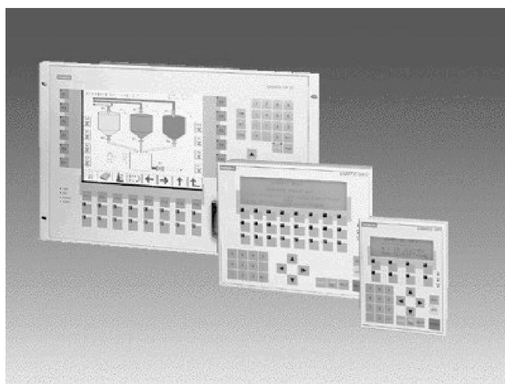


图 2-8　人机界面

第四节　SIMATIC 工业软件

一、STEP 7

STEP 7 是一种编程和组态软件。使用该软件，可对自动化系统 SIMATIC S7-300 和 S7-400

进行编程和组态。包括采用数字量模块对 CPU 进行硬件配置，连接 PROFIBUS、以太网组件以及其他网络组件。该编程软件符合 IEC 61131 标准。

STEP 7 的特点：通用数据存储，项目结构简明；采用 STL、LAD 和 FBD 语言编程，各编程语言之间可以切换；CPU 属性可调；模块地址可调；可显示模块诊断信息，读取错误消息缓存器；可显示交叉参考和参考数据。

STEP 7 如图 2-9 所示。

图 2-9　STEP 7

图 2-10　S7-GRAPH

二、顺序控制编程软件 S7-GRAPH

使用工程软件 S7-GRAPH，可对顺序控制系统进行组态、调试和编程，该软件符合标准 IEC 61131-3，通过图形化连线代替昂贵的编程。可按步和转换（步之间的转换）的顺序，对过程进行概览分析。之后，还可对步的内容进行编程。步之间的转换也可使用 LAD 或 FBD 语言进行编程。使用该软件，可对整个编程方案进行概览。另外，该软件还提供有各种诊断功能。

S7-GRAPH 如图 2-10 所示。

三、状态控制编程软件 S7-HiGRAPH

使用 S7-HiGRAPH，可借助状态图进行非同步过程的图形化描述。使用该软件，可图形化描述过程状态以及可能的状态转换。可任意放置的图像元素，提高了软件使用灵活性。通过简便的集成监控和消息功能，用户可以很容易地分析错误，减少停机时间。使用状态图，既可描述自动操作，也可描述手动操作。而且不仅适用于 PLC 编程人员，还适用于机器制造商（示教）及调试和维修工程师。

S7-HiGRAPH 适用于自动化系统 SIMATIC S7-300（建议使用 CPU 315 或以上）、SIMATIC S7-400、SIMATIC C7（建议使用 C7-626 或以上）以及 SIMATIC WinAC。

S7-HiGRAPH 如图 2-11 所示。

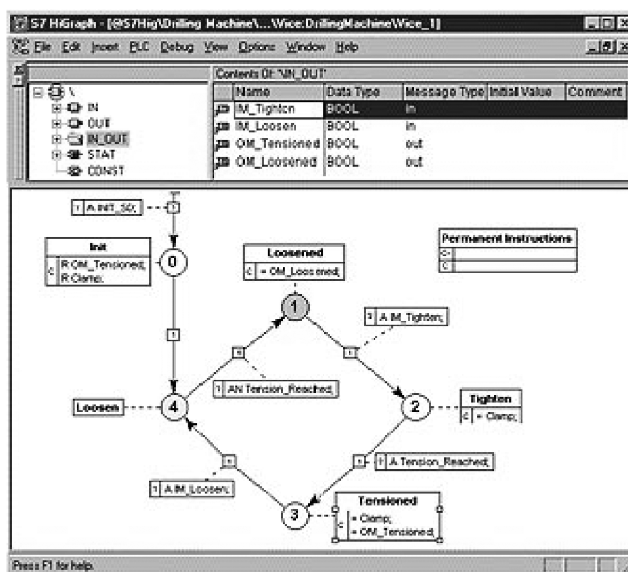


图 2-11　S7-HiGRAPH

四、高级编程语言 S7-SCL

S7-SCL（Structured Control Language，结构化控制语言）是基于 PASCAL 的高级语言，用于存储程序

```
SCL: Programming S7 Blocks - [Operate]
File Edit Insert PLC Debug View Options Window Help

FUNCTION_BLOCK FB27
VAR_INPUT
  SIG_SEL      : INT   := 0;
  GRP1_SEL     : BOOL  := 0;
  GRP2_SEL     : BOOL  := 0;
  GRP3_SEL     : BOOL  := 0;
END_VAR

VAR_OUTPUT
  SEL_OUT      : INT   := 0;
  GRP1_OUT     : BOOL  := 0;
  GRP2_OUT     : BOOL  := 0;
  GRP3_OUT     : BOOL  := 0;
END_VAR

VAR
  SELECT       : INT;
  MAX          : INT;
END_VAR

BEGIN
  SELECT       := SIG_SEL;
  MAX          := 3;
  IF SELECT  <  0 THEN               //make it positive
    SELECT     := -SELECT;
  END_IF;
  IF SELECT  >  MAX THEN             //limit to MAX
    SELECT     := MAX;
  END_IF;
  SEL_OUT      := SELECT;

Press F1 for help.                    Insert   1:1   offline
Start  DateManager - [D...  OnTime  SCL: Programmi...   11:36
```

图 2-12　S7-SCL

控制的编程。S7-SCL 有 PLCOpen Base Level 证书。使用 S7-SCL 具有的优点：简单、快速的程序创建；高质量的 PLC 程序；更佳的可懂度；更简便的调试 。用户可以为自动化任务构建省时的高性价比解决方案。该软件可用于所有自动化系统 SIMATIC S7-300（建议使用 CPU 314 或以上）、SIMATIC S7-400、SIMATIC C7 以及 SIMATIC WinAC。

S7-SCL 如图 2-12 所示。

五、SIMATIC WinAC Basis

SIMATIC WinAC Basis 是一种基于 PC 的控制解决方案，适用于解决小型控制任务以及典型的 PC 任务，具有较高的性价比。SIMATIC WinAC Basis 具有控制功能和技术功能以及用于可视化、数据处理和通信的标准应用程序。软件运行环境为 Windows NT 操作系统。对于苛刻和具有严格实时要求的任务，可采用 SIMATIC WinAC RTX。它可直接安装在 Windows NT 下，其增强的实时性能保证对控制部分具有确定性响应。

方便的 SIMATIC WinAC 如图 2-13 所示。

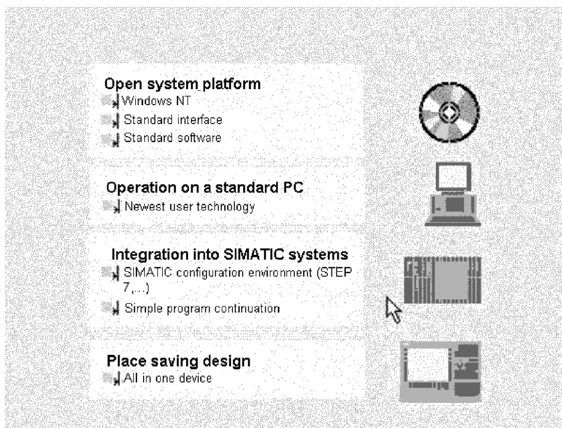


图 2-13　方便的 SIMATIC WinAC

SIMATIC WinAC 组件如下：

1. Controlling 组件

软 PLC 的操作与硬 PLC（例如 CPU 315-2 DP）相同。使用标准工具 SIMATIC Manager 进行编程和诊断（LAD/FBD/STL Editor，Monitor/ModifyVariable）。通过优先级控制，可简便地调整软 PLC 的性能（从 Windows NT 实时响应，到 Windows NT 中的辅助应用程序）。软 PLC 通过密码保护来保证安全，防止影响控制。永久性数据可防止系统故障。

使用 SIMATIC WinAC 数据元素，可创建和显示 B&B（操作与观测）操作员界面，并可与 SoftContainer 一起显示。也可使用 VB 实现 HMI 操作员界面。也可使用 ProTool/Pro 进行快速 HMI 组态。

2. Computing 组件

PLC 的数据交换由 PC 完成。数据元素也由 PC 处理（SIMATIC Data Control）。使用这些数据元素，可简便地连接 Windows 对象（OCX 组件）与软 PLC（例如存储位字，输入和输出等）。

使用数据元素，也可通过普通的办公网络（以太网）与 PLC 进行数据交换（仅通过 SIMATIC S7 416-2 DP ISA）。

3. Windows 逻辑控制器

Windows 逻辑控制器（Windows Logic Controller，WinLC）是 CPU 的软件解决方案，如图 2-14 所示。可在安装有 WinLC 的 PC 上，显示 CPU 的功能。WinLC 程序可仿真 CPU 315-2 DP 的整个操作，也可根据 CPU 315-2 DP 定制。故障 LED 或操作类型开关的排布都和硬件 CPU 相对应。WinLC 可处理 1024 点数字量输入和 128 点模拟量输入。因此，可通过一个分布式 I/O（如 ET 200M）连接外部设备。

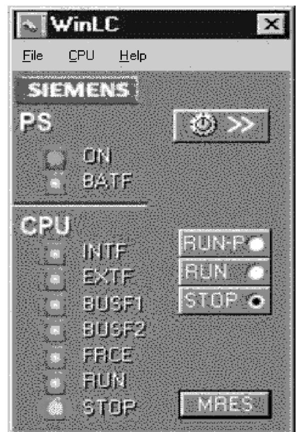


图 2-14　WinLC

控制性能则取决于处理器和 RAM 的性能以及 WinLC 软件的可调循环时间。WinLC 的循环时间可根据具体的控制任务设置。一个 WinLC 循环包括读取过程映像输入表中的输入、程序执行、生成过程映像输入表（例如等待时间的执行），直到规定的最小循环时间到。在剩余时间内，Windows NT 完成其他当前任务。

实时解决方案则形成了插槽式 PLC。插槽式 PLC 是一个应用程序，相当于 CPU 416-2 DP 中的一个功能包，可实现确定性的响应，并且响应时间短，与操作系统 Windows NT 无关。操作系统可在插槽式 PLC 运行时起动。

六、SIMATIC ProTool/Pro

使用 ProTool 和/或 ProTool/Pro，可对 HMI 进行组态。ProTool 可将构思方案简便、快速地转换为用于可视化系统的清晰图像。其优点是可以利用 Windows 环境，即可使用标准图形处理程序来创建图形连接。可以将 ProTool 集成到 SIMATIC Manager 中，并在这里处理符号表。组态软件也可作为独立版使用。

OP 3、OP 7 和 OP 17 型操作员面板可使用软件 ProTool/Lite 进行组态。图形化操作员面板 OP 27、OP 37、TP 27 和 TP 37 需使用软件 ProTool。基于 Windows 的系统（如 OP 37 Pro）可使用 ProTool/Pro 进行组态。

ProTool/Pro 如图 2-15 所示。

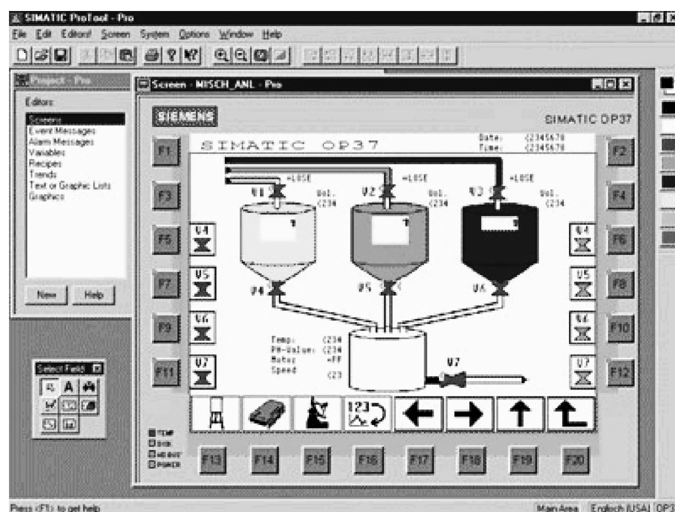


图 2-15　ProTool/Pro

七、HMI SIMATIC WinCC

SIMATIC WinCC 表示“Windows 控制中心”。WinCC 是一个基于 Windows NT 或 Windows 95 的高性能过程可视化系统。该 32 位系统提供了优先多任务处理功能，以便能够快速和有效地对过程事件和报警做出响应。通过 WinCC，可以顺利地集成到现有自动化系统中。它采用一个与多个标准应用程序共用的数据库，可与其他 Windows 应用程序或其他 SIMATIC 组件进行数据交换。WinCC 中还集成有用于对图形元素进行简便组态的对象库。

其属性如下：

1）用于方便地创建过程映像的图形化编辑器。

2）带有归档系统的消息系统。

3）测量值归档。

4）报告系统。

5）用于组态基本功能的选项。

6）通用数据库。

WinCC 如图 2-16 所示。

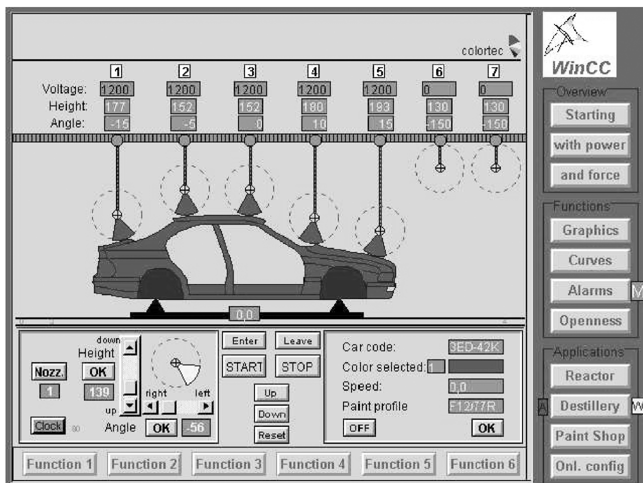


图 2-16 WinCC

八、PCS 7 过程控制系统

PCS 7 是 TIA 自动化概念中的过程控制系统。它基于采用经过挑选的便利组件而作为控制系统运行。另外，它还拥有一些可确保通过 I/O 控制和人机界面进行工程组态时的特殊系统性能的功能扩展，还提供了一个用于可满足过程仪表和控制系统的苛刻要求的功能包。

其属性有：定义的起动和重新起动，操作概念，控制系统消息概念，访问授权，使用寿命监测，实时同步，安全过程控制，包含有预制功能块的库，用户友好的组态，用于非连续过程的附加软件包。

PCS 7 如图 2-17 所示。

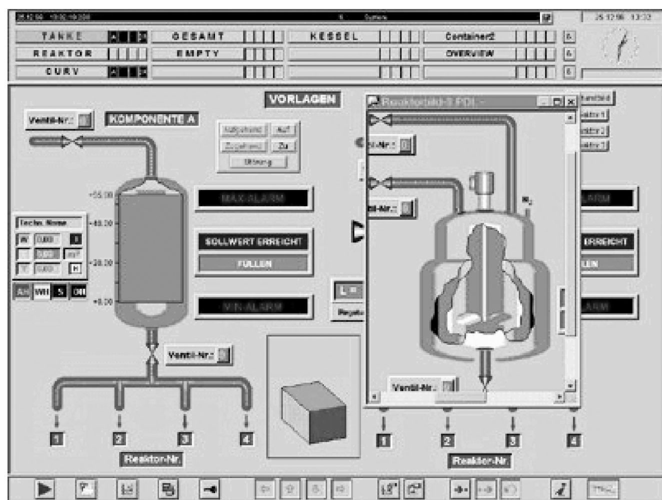


图 2-17 PCS 7

第五节 驱 动 技 术

不管是标准传动还是大型传动，不管是交流传动还是直流传动，变速驱动器都可用于全集成自动化以及 SIMATIC 环境中。其具有以下优点：从控制级到驱动器的统一性和畅通性；DRIVE 工程师站可完全集成在 STEP 7 环境中；SIMATIC Manager 用于通用数据管理；显著降低驱动技术在自动化解决方案中的集成时间。

在 TIA 框架内提供有以下驱动系统和驱动组件，用于各种不同的应用。

一、低压电动机

业界首款专为机器和系统配置打造的高性能交流电动机（见图 2-18），一种前瞻未来的解决

方案：免维护，高动态性能。

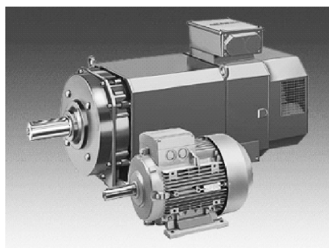


图 2-18　低压电动机

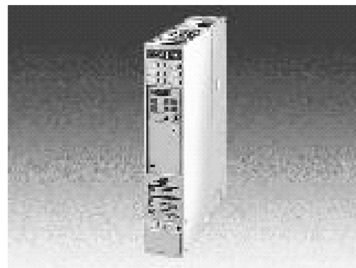


图 2-19　SIMOVERT MASTERDRIVES 变频器

二、SIMOVERT MASTERDRIVES 变频器

使用 SIMOVERT MASTERDRIVES 变频器（见图 2-19），可使交流电动机更好地变速运行。该款变频器全球通用，适用于 230～690V 各种电源电压，且已在全球广泛使用。

三、标准变频器

MICROMASTER 和 MICRO-/MIDIMASTERVectors 均是高性能的变频器，功率范围介于75～120kW 之间，结构紧凑。凭其无编码器矢量控制功能，可用于苛刻的中端应用。

COMBIMASTER 结构紧凑，集成有 DS 低压电动机和变频器。

MICROMASTER Integrated 是一款可直接集成在不同制造商的 DS 低压电动机中的变频器（防护等级 IP 65）。

MICRO-/MIDIMASTER Ecos 变频器用于供热、通风和空调行业。

标准变频器如图 2-20 所示。

四、SIMOREG 直流调速器

SIMOREG 直流调速器是一种三相交流电源直接供电的全数字控制装置，其结构紧凑，用于可调速直流电动机电枢和励磁供电。装置额定电流范围为 15～2000A，并可通过并联 SIMOREG 整流装置进行扩展。该装置应用于起重设备、滑雪缆车、电梯、起重机、飞剪和其他反转驱动装置。

SIMOREG 直流调速器如图 2-21 所示。



图 2-20　标准变频器



图 2-21　SIMOREG 直流调速器

第三章　S7-300/400 PLC的硬件配置

通过本章的学习主要了解和掌握西门子 PLC S7-300 和 S7-400 系列的组成及功能模块。

第一节　S7-300 的基本组成

S7-300 属于模块式 PLC，主要由机架、CPU 模块、信号模块、功能模块、接口模块、通信处理器、电源模块和编程设备组成，其具体构成如图 3-1 所示。

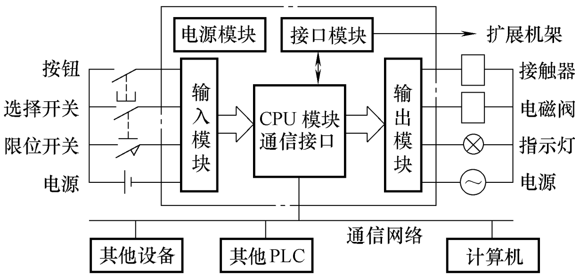


图 3-1　系统构成

S7-300 是模块化中小型 PLC，适用于中等控制要求，而 S7-400 则是具有中高档性能的 PLC，采用模块化无风扇设计，适用于对可靠性要求极高的大型复杂系统的控制。S7-300/400 的模块化组成方式，极大地方便和满足了各个领域的控制任务，用户可以根据控制系统的具体要求，来选择不同的控制模块，当系统出现故障时，也可以非常方便地更换模块，方便了系统的维修。

一、S7-300 的概况

S7-300 一般包括电源模块（PS）、CPU、信号模块（SM）、功能模块（FM）、接口模块（IM）和通信处理器（CP）模块等。S7-300 的 CPU 模块（简称为 CPU）都有一个编程用的 RS-485 接口，有的有 PROFIBUS-DP 接口或 PtP 串行通信接口，可以建立一个 MPI（多点接口）网络或 DP 网络，CPU 采用智能化的诊断系统连续监控系统的功能是否正常并记录错误和特殊系统事件。功能最强的 CPU 的 RAM 为 512KB，最大有 8192 个存储器位，512 个定时器和 512 个计数器，数字量最大为 65536，模拟量通道最大为 4096，计数器的计数范围为 1 ~ 999，定时器的定时范围为 10ms ~ 9990s，有 350 条指令。

S7-300 中还有看门狗中断、过程报警、日期时间中断和定时中断等功能。S7-300 已经将 HMI 服务集成到操作系统内部，大大降低了人机对话编程的难度。

二、S7-300 的系统结构

S7-300 采用紧凑和无槽位限制的模块结构，将电源模块、CPU、信号模块、功能模块、接口模块和通信处理器等安装在导轨上。轨道为一种专门的金属机

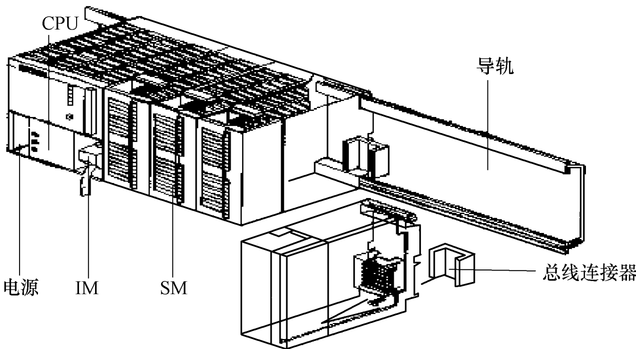


图 3-2　S7-300 的安装

架，只需要将模块挂在 DIN 标准的安装轨道上，用螺钉锁紧就可以了。有很多种不同长度规格的导轨供用户选择。S7-300 的安装如图 3-2 所示。

电源模块总是安装在机架的最左边，CPU 模块紧紧靠近电源模块，如果还要安装接口模块，则把接口模块安装在 CPU 模块的右边。S7-300 用背板总线将除电源模块之外的各个模块连接起来，背板总线集成在模块上，模块通过 U 形总线连接器相连接，每个模块都有一个总线连接器，后者插在各模块的背后。安装时先将总线连接器插在 CPU 模块上，并固定在导轨上，然后依次装入各个模块。外部接线接在信号模块和功能模块的前连接器的端子上，前连接器用插接的方式安装在模块前门后面的凹槽中。

每个轨道最多只能安装 8 个信号模块、功能模块和通信处理器模块。当系统需要大于 8 个模块时，则可以增加扩展机架，如图 3-3 所示。除了带 CPU 的中央机架（CR）外，最多可以增加 3 个扩展机架（ER），每个机架可以插入 8 个模块（不包括电源模块、CPU 和接口模块），4 个机架最多可以安装 32 个模块。

机架最左边是 1 号槽，最右边是 11 号槽，电源模块总是安装在 1 号槽中，中央机架的 2 号槽安装 CPU 模块，3 号槽安装的是接口模块，这 3 个槽号被固定占用，不能安装其他模块，其他模块只能安装在 4～11 号槽中。

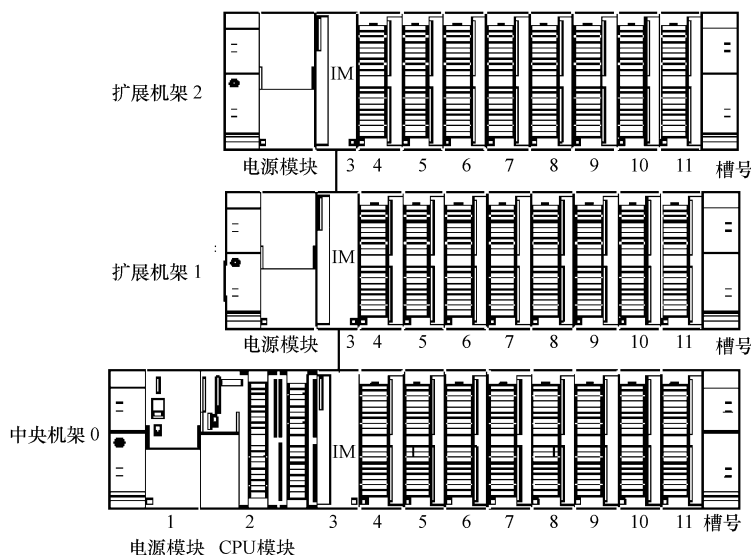


图 3-3　多机架 PLC（S7-300）

S7-300 的模块是通过总线连接器连接的，各个槽号是相对的，在机架轨道上不存在物理槽位。当某个槽位不使用时，例如 5 号槽位上没有插任何模块，而 4 号槽位插有功能模块，6 号槽位上插有信号模块，虽然 5 号槽位没有使用，占用了一个槽号位，但在物理上，6 号槽位和 4 号槽位的模块是连在一起的。

当需要扩展机架时，把接口模块插入 3 号槽，负责与其他扩展机架进行数据通信。如果只需扩展一个机架，可以选择价格比较便宜的 IM365 接口模块，两个接口模块用 1m 长的固定电缆连接。但采用 IM365 接口模块有个非常不利的弊端就是 IM365 接口模块不能给扩展机架提供通信总线，所以扩展机架上只能安装信号模块，不能安装除通信模块外的其他智能模块。扩展机架电源由 IM 265 提供，两个机架的 DC 5V 电源的总电流应该在 PLC 允许的范围内。

当需要扩展 3 个机架时，可以使用 IM360/361 接口模块，中央机架使用 IM360，扩展机架使用 IM361，各相邻机架之间的电缆最长为 10m，每个 IM361 需要一个外部 24V 电源来对所有的扩展机架上的模块进行供电，可以通过电源连接器连接 PS307 负载电源，所有的扩展模块均可以安装在扩展机架上。每个机架上的模块除了安装的信号模块、功能和通信处理器模块不能超过 8 模块外，还要受到背板总线 DC 5V 供电电流的限制。中央机架上的 DC 5V 电源由 CPU 产生，其额定电流值与 CPU 的型号有关，扩展机架的背板总线的 DC 5V 电源由接口模块 IM361 产生。关于扩展机架的详细配置与说明见第三章第五节 S7-300/400 扩展机架的配置与说明。

三、S7-300 模块诊断与过程诊断

1. 模块的诊断功能

S7-300 中有的信号模块具有对信号进行监视（诊断）和过程中断的功能。通过诊断可以确定数字量模块获取的信号是否正确，或者模拟量模块的处理是否正确。

数字量输入/输出模块可以诊断出以下故障：无编码器电源、RAM 故障、过程报警丢失等。模拟量输入模块可以诊断出无外部电压、共模故障、组态/参数错误、断线和测量范围上溢出或下溢出等故障。模拟量输出模块可以诊断出无外部电压、组态/参数错误、断线和对地短路等故障。

2. 过程中断

过程中断可以对过程信号进行监视和响应，根据设置的参数，可以选择数字量输入模块的每个通道组是否在信号的上升沿、下降沿产生过程中断，或在两个边沿都产生过程中断。信号模块可以对每个通道的一个中断进行暂存。

模拟量输入模块通过上限值和下限值定义一个工作范围，模块将测量值与上、下限值进行比较。如果超过限值，则执行过程中断。执行过程中断时，CPU 暂停执行用户程序，或暂停执行低优先级的中断程序，来处理相应的诊断中断功能模块（OB40）。

3. 状态与故障显示 LED

1）SF（系统出错/故障显示，红色）：CPU 硬件故障或软件错误时亮。

2）BATF（电池故障，红色）：电池电压低或没有电池时亮。

3）DC5V（+5V 电源指示，绿色）：5V 电源正常时亮。

4）FRCE（强制，黄色）：至少有一个 I/O 被强制时亮。

5）RUN（运行方式，绿色）：CPU 处于 RUN 状态时亮；重新起动时以 2Hz 的频率闪亮；HOLD（单步、断点）状态时以 0.5Hz 的频率闪亮。

6）STOP（停止方式，黄色）：CPU 处于 STOP、HOLD 状态或重新起动时常亮。

7）BUSF（总线错误，红色）。

4. 模式选择开关

1）RUN—P（运行—编程）位置：运行时还可以读出和修改用户程序，改变运行方式。

2）RUN（运行）位置：CPU 执行、读出用户程序，但是不能修改用户程序。

3）STOP（停止）位置：不执行用户程序，可以读出和修改用户程序。

4）MRES（清除存储器）：不能保持。将钥匙开关从 STOP 位置搬到 MRES 位置，可复位存储器，使 CPU 回到初始状态。

5）复位存储器操作：通电后从 STOP 位置扳到 MRES 位置，“STOP” LED 熄灭 1s，亮 1s，再熄灭 1s 后保持亮。放开开关，使它回到 STOP 位置，然后又回到 MRES 位置，“STOP” LED 以 2Hz 的频率至少闪动 3s，表示正在执行复位，最后“STOP” LED 一直亮。

第二节 S7-300 的功能模块

一、S7-300 的 CPU

在介绍 CPU 性能之前，首先了解一下常用术语。

1. 存储器

1）装载存储器（Load Memory）：其用途是装载用户程序。用户程序经由通信接口，从编程设备传送到 S7-300 CPU 的装载存储器中。

2）工作存储器（Work Memory）：物理上为 CPU 内置 RAM 的一部分，当 CPU 处于运行状态时，用户程序和数据从装载存储区调入工作存储区，在工作存储器中运行。

3）系统存储器（System Memory）：S7-300 将 CPU 的一部分内置 RAM 划分出来，用于位存储、I/O 映像寄存器、计数器、定时器等。系统存储器与工作存储器同属于 CPU 集成的物理内存，用户程序代码和数据均在这两部分存储区中执行。

4）微存储器卡 MMC（Micro Memory Card）：用于对装载存储器的扩充，CPU 模块上有专用的 MMC 插槽，MMC 可拆卸，最大容量的 MMC 为 8MB。作为装载存储器，MMC 用于对用户程序和数据的断电保护，也可存储 S7-300 系统程序以利于以后的系统升级。

2. CPU

S7-300 有多种不同型号的 CPU，这些 CPU 按性能等级划分，几乎涵盖了各种应用范围。从目前的情况来看，大体有 4 个系列。

1）标准型 CPU 系列。包括：CPU 313、314、315、315-2 DP、316-2 DP、318-2。型号尾部有后缀“DP”字样的，表明该型号 CPU 集成有现场总线 PROFIBUS-DP 通信接口。此外还有几种重新定义型的 CPU，包括 CPU 312、314、315-2 DP、317-2 DP 等。

2）集成型 CPU 系列。主要有 CPU 312 IFM 和 CPU 314 IFM 两种。在这两种 CPU 内部集成了部分 I/O 点、高速计数器及某些控制功能。

CPU 312 IFM 集成的特殊功能有：1 个 4 输入端高速计数器，计数器长度为 32 位（含符号位），计数频率可达 10kHz；1 个频率测量通道，最高可测 10kHz，采样周期可调为 0.1s、1s、10s；过程中断功能等。这些功能的实现，需要在编程时对内部集成的数字输入通道（124.6 ~ 125.1）进行专门定义。

CPU 314 IFM 集成的功能有：1 个 4 输入端高速计数器（也可定义为两个 2 输入端计数器），计数频率可达 10kHz；1 个频率测量通道，最高可测 10kHz，采样周期可调为 0.1s、1s、10s；开环定位功能，通过一个 24V 增量编码器进行位置测量，需要占用 3 个数字输入端；过程中断功能；PID 控制功能，可实现闭环控制。上述涉及数字量输入的功能，同样需要对内部集成的数字输入端（126.0 ~ 126.3）进行专门定义。

3）紧凑型 CPU 系列，型号后缀带有字母 C，包括 CPU 312C、313C、313C-2 PtP、313C-2 DP、314C-2 PtP、314C-2 DP。型号尾部有后缀“PtP”字样的，表明该型号 CPU 集成有第二个串行口，两个串行口都有点对点（PtP）通信功能。

CPU 313C、CPU 314C-2 PtP、CPU 314C-2 DP 三种型号中集成有模拟量输入/输出，其中 4 路输入的规格为 DC ±10V、0 ~ 10V、±20mA、4 ~ 20mA，分辨率为 11 位 + 符号位，积分时间可调为 2.5ms、16.6ms、20ms。另有一路模拟量输入可测 0 ~ 600Ω 的电阻，或接 Pt100 热电阻。

两路模拟量输出规格为：DC ±10V、0 ~ 10V、±20mA、4 ~ 20mA，各通道转换时间为 1ms。

CPU 312C 集成的特殊功能有：2 通道高速计数器，最大频率为 10kHz；2 通道频率测量，可

测最大频率为 10kHz；2 通道脉冲宽度调制输出，最高输出频率为 2.5kHz。

CPU 313C、313C-2 PtP、313C-2 DP 集成的特殊功能有：3 通道高速计数器，最大频率为 30kHz；3 通道频率测量，可测最大频率为 30kHz；3 通道脉冲宽度调制输出，最高输出频率为 2.5kHz；PID 闭环控制。

CPU 314C-2 PtP、314C-2 DP 集成的特殊功能有：4 通道高速计数器，最大频率为 60kHz；4 通道频率测量，可测最大频率为 60kHz；4 通道脉冲宽度调制输出，最高输出频率为 2.5kHz；1 路位置控制；PID 闭环控制。

4）故障安全型 CPU 系列，是西门子公司最新推出的具有更高可靠性的 CPU，主要型号有 CPU 315F、317F-2 DP。表 3-1 列出了几种 CPU 的主要技术参数。

表 3-1　几种 CPU 的主要技术参数

CPU 型号	313	315-2DP（标准型）	315-2DP（新型）	314C-2PtP	314C-2DP	312 IFM
装载存储器	20KB RAM 4MB MMC	96KB RAM 4MB MMC	8MB MMC	4MB MMC	4MB MMC	20KB RAM/ 20KB ROM
内置 RAM	12KB	64KB	128KB	48KB	48KB	6KB
浮点数运算时间	60μs	50μs	6μs	15μs	15μs	60μs
最大 DI/DO	256	1024	1024	992	992	256
最大 AI/AO	64/32	256 /128	256	248/124	248/124	64/32
最大配置 CR/ER	1/0	1/3	1/3	1/3	1/3	1/0
定时器	128	128	256	256	256	64
计数器	64	64	256	256	256	32
位存储器	2048B	2048B	2048B	2048B	2048B	1024B
通信接口	MPI 接口	MPI 接口，DP 接口	MPI/PtP 接口，DP 接口	MPI/PtP 接口，PtP 接口	MPI 接口，DP 接口	MPI 接口

二、S7-300 的数字量模块

S7-300 的数字量模块基本为三大类：SM321 数字量输入模块、SM322 数字量输出模块、SM323 数字量输入/输出模块。

1. SM321 数字量输入模块

根据输入点数的多少，可分为 8 点、16 点、32 点三类。输入电压类型有直流和交流两种，电压等级有 DC24V、DC48 ~ 125V、AC120V、AC120/230V 等几种。例如 SM321：DI 32 × 24VDC，为 32 点输入、直流 24V 信号模块；SM321：DI 16 × 120VAC，为 16 点输入、交流 120V 信号模块；SM321：DI 16 × 24VDC，Interrupt，为 16 点输入、直流 24V、带硬件故障诊断及中断的信号模块。

信号模块输入点通常要分成若干组，每组在模块内部有电气公共端，选型时要考虑外部开关信号的电压等级和形式，不同电压等级的信号必须分配在不同的组。模块输入点的分组情况要查阅相关的技术手册，限于篇幅，这里不再给出。

2. SM322 数字量输出模块

SM322 系列有 32 点、16 点、8 点三种类别，输出开关器件有晶体管输出方式、晶闸管输出方式、继电器输出方式。负载电压等级有 DC24V、DC48 ~ 125V、AC120V、AC120/230V 等几种。例如 SM 322：DO8 × 230VAC/5 A REL.，为 8 点继电器输出、最大可带 AC230V/5A 的负载。

3. SM323 数字量输入/输出模块

这一类型的模块目前有两种：DI 16/DO 16×24V DC/0.5A 和 DI 8/DO 8×24VDC/0.5A。前者有 16 个数字输入点和 16 个数字输出点，16 个输入点为 1 组，内部共地；16 个输出点分成两组，两组的内部结构相同，均为晶体管输出，每 8 个输出点共用一对负载电源端子。后者为 8 输入/8 输出模块，输入/输出均为 1 组，内部结构与前者相同。

三、S7-300 的模拟量模块

S7-300 的模拟量输入/输出模块包括：SM 331 模拟量输入（AI）系列、SM 332 模拟量输出（AO）系列、SM 334 模拟量输入/输出（AI/AO）系列。

1. SM 331 系列模拟量输入（AI）模块

SM 331 模拟量输入模块的核心部件是 A-D 转换器，由于若干通道合用一个 A-D 转换器，所以在模拟量进入 A-D 转换器之前，需要有多路模拟转换开关来选择通道，各通道是循环扫描的，因此每一个通道的采样周期不仅取决于各通道的 A-D 转换时间，还取决于所有被激活的通道数量。为了尽量缩短扫描周期，加大采样频率，有必要利用 STEP 7 编程软件屏蔽掉那些不用的通道。

常用的 SM 331 系列模拟量输入模块有 5 种，表 3-2 给出了这 5 种模块的主要技术指标。

表 3-2　常用模拟量输入模块的主要技术指标

技术指标	AI 8×12 Bit	AI 8×16 Bit	AI 2×12 Bit	AI 8×RTD	AI 8×TC
输入点数/组数	8 点/4 组	8 点/4 组	2 点/1 组	8 点/4 组	8 点/4 组
分辨率	9 位+符号位 12 位+符号位 14 位+符号位	15 位+符号位	9 位+符号位 12 位+符号位 14 位+符号位	15 位+符号位	15 位+符号位
技术指标	AI 8×12 Bit	AI 8×16 Bit	AI 2×12 Bit	AI 8×RTD	AI 8×TC
测量方式	电流、电压、电阻器、温度计	电流 电压	电流、电压、电阻器、温度计	电阻器 温度计	温度计
测量范围选择	任意	任意	任意	任意	任意

A-D 转换的结果按 16 位二进制补码形式存储，即占用 1 个字（两个字节）的长度。最高位为符号位，为“1”表示转换结果为负值，为“0”表示结果为正值。当转换精度不够 15 位时（例如 9 位+符号位），有效位（包括符号位）从高字节的最高位开始向下排，无效位补零。表 3-3 为三种转换精度的数据存储格式，S 位为符号位，标有×的位被补为 0。

表 3-3　A-D 转换结果存储格式示例

分辨率	高　字　节								低　字　节							
15 位+符号位	S	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
12 位+符号位	S	1	0	1	0	1	0	1	0	1	0	1	0	×	×	×
9 位+符号位	S	1	0	1	0	1	0	1	0	1	×	×	×	×	×	×

2. SM332 系列模拟量输出（AO）模块

模拟量输出模块用于将 S7-300 的数字信号转换为系统所需要的模拟量信号，控制模拟量调节器、执行机构或者作为其他设备的模拟量给定信号，其核心部件为 D-A 转换器。

SM332 系列模块的输出精度主要有 12 位和 16 位两种；输出通道主要有 2 通道和 4 通道两种形式；输出信号可为电压或电流。电压的输出范围可调为：1～5V、0～10V、±10V。电流的输出范围可调为：0～20mA、4～20mA、±20mA。

例如模块 SM332；AO 4×12Bit，共有 4 个通道，每个通道的分辨率均为 12 位，可分别设置为电流输出或电压输出。电流输出为两线式，电压输出可为两线式，也可为四线式，采用四线式时，其中两个端子的引出线用于测量负载两端的电压，这样可以提高电压的输出精度。

3. SM334 系列模拟量输入/输出（AI/AO）模块

SM334 模拟量 I/O 模块主要有 AI4/AO 2×8/8 Bit 和 AI4/AO2×12 Bit 两种规格。

AI4/AO2×12 Bit 模块的特点是：分辨率为 12 位；4 个输入通道分为两组，可以测量电压信号、电阻器、热电耦，电压信号的测量范围是 0～10V，不能测量电流信号；两路电压模拟量输出，输出范围为 0～10V，两线制接法，没有测量端。

AI4/AO2×8/8 Bit 模块的特点是：分辨率为 8 位；4 路输入可测量电压和电流信号，两路输出；输入与输出的范围均为：电压 0～10V，电流 0～20mA；输入/输出形式的选择不是通过软件组态，而是通过接线形式来确定，该点是与其他模块的最大不同之处。

四、S7-300 的电源模块

PS 307 电源模块有 2A、5A 和 10A 三种规格，将输入的单相交流电压（120/230V，50/60Hz）转变为直流 24V 提供给 S7-300 PLC 使用，同时也可作为负载电源，通过 I/O 模块向使用 DC24V 的负载（如传感器、执行机构等）供电。PS 307 电源模块的输入与输出之间有可靠的隔离。如果正常输出额定电压 24V，面板上的绿色 LED 灯点亮；如果输出电路过载，LED 灯闪烁，输出电压下降；如果输出短路，则输出电压为零，LED 灯灭，短路故障解除后自动恢复。另外，LED 灯灭的状态下，也有可能是输入交流电源电压低所致，此时模块自动切断输出，故障解除后自动恢复。

图 3-4 是 PS 307 电源模块原理图。图中 L+ 和 M 端子为 DC24V 的正、负输出，各提供两个接线端子以利于分别向 CPU 模块和 I/O 模块接线。L1 和 N 为交流电源输入端子。

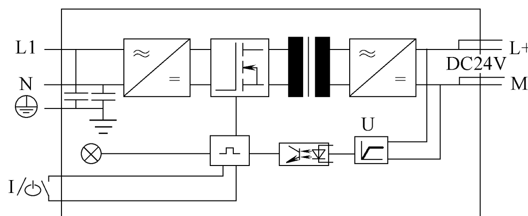


图 3-4 PS 307 电源模块原理图

五、数字量的 I/O 编址

S7-300 的数字 I/O 地址由地址标识符、地址的字节部分和位部分组成，地址标识符 I 表示输入，Q 表示输出，例如 I0.7 是一个输入数字量的地址，表示 0 号字节的第 7 位。

S7-300 对各个 I/O 点的编址是依据其所属模块的安装位置决定的，依据规定，各种信号模

槽位号	1	2	3	4	5	6	7	8	9	10	11
机架 0	PS	CPU	IM	0.0～3.7	4.0～7.7	8.0～11.7	12.0～15.7	16.0～19.7	20.0～23.7	24.0～27.7	28.0～31.7
机架 1	PS		IM	32.0～35.7	36.0～39.7	40.0～43.7	44.0～47.7	48.0～51.7	52.0～55.7	56.0～59.7	60.0～63.7
机架 2	PS		IM	64.0～67.7	68.0～71.7	72.0～75.7	76.0～79.7	80.0～83.7	84.0～87.7	88.0～91.7	92.0～95.7
机架 3	PS		IM	96.0～99.7	100.0～103.7	104.0～107.7	108.0～111.7	112.0～115.7	116.0～119.7	120.0～123.7	124.0～127.7

图 3-5 S7-300 数字量 I/O 模块的默认地址

块应安装在 4 号至 11 号槽位。因此，CPU 从 4 号槽位开始为 I/O 模块分配地址，每个槽位所占用的 I/O 地址是系统默认的，以字节为单位。

图 3-5 给出了各个机架和槽位的数字量 I/O 模块的默认地址。

图中，每个槽位最多 32 个点，占 4 个字节。举例来说，若中央机架（机架 0）的 4 号槽位上安装了 SM 321：DI 32 × 24VDC 模块，则该模块上的 32 个数字输入点地址依次为：I0.0 ~ I0.7、I1.0 ~ I1.7、I2.0 ~ I2.7、I3.0 ~ I3.7。若为数字输出模块，例如 SM 322：DO 32 × 24VDC/0.5A，则 32 个输出点地址依次为：Q0.0 ~ Q0.7、Q1.0 ~ Q1.7、Q2.0 ~ Q2.7、Q3.0 ~ Q3.7。如果不是 32 点的模块，例如 SM 321：DI 16 × 24VDC，则各点地址依次为：I0.0 ~ I0.7、I1.0 ~ I1.7，后面的 I2.0 ~ I2.7 不能用。

六、其他功能模块

1. 计数器模块

模块的计数器均为 0 ~ 32 位或 ± 31 位加减计数器，可以判断脉冲的方向，模块给编码器供电，达到比较值时发出中断，可以 2 倍频和 4 倍频计数，有集成的 DI/DO。FM 350-1 是单通道计数器模块，可以检测最高达 500kHz 的脉冲，有连续计数、单向计数、循环计数 3 种工作模式。FM350-2 和 CM 35 都是 8 通道智能型计数器模块。

2. 位置控制与位置检测模块

FM 351 双通道定位模块用于控制变级调速电动机或变频器国。FM 353 是步进电动机定位模块。FM 354 是伺服电动机定位模块。FM357 可以用于最多 4 个插补轴的协同定位。FM 352 是高速电子凸轮控制器，它有 32 个凸轮轨迹，13 个集成的 DO，采用增量式编码器或绝对式编码器。SM 338 超声波传感器用于位置检测，具有无磨损、保护等级高、精度稳定不变等特点。

3. 闭环控制模块

FM 355 闭环控制模块有 4 个闭环控制通道，有自优化温度控制算法和 PID 算法。

4. 称重模块

SIWAREX U 称重模块是紧凑型电子秤，测定料仓和贮斗的料位，对吊车载荷进行监控，对传送带载荷进行测量或对工业提升机、轧机超载进行安全防护等。SIWAREX M 称重模块是有校验能力的电子称重和配料单元，可以组成多料称系统，安装在易爆区域。

第三节　S7-400 系统简介

SIMATIC S7-400 是用于中、高档性能范围的 PLC，模块化及无风扇的设计，坚固耐用，容易扩展和通信能力强，容易实现的分布式结构以及友好的用户操作。SIMATIC S7-400 成为中、高档性能控制领域中首选的理想解决方案，其具有以下特征：

1）性能分级的 CPU 平台。

2）向上兼容 CPU。

3）具有更方便接线的端子系统。

4）最佳的通信和网络选择。

5）更方便的操作员接口系统。

6）为所有模块分配软件参数。

7）无需风扇便能进行工作。

S7-400 系统由以下组件组成：机架、电源模块（PS）、S7-400 CPU、信号模块（SM）、通信处理器（CP）、功能模块（FM）等。S7-400 对模块数量限制的上限远远大于 S7-300，因而有极

强的扩展能力。信号模块的更换可以热插拔，而不必暂停生产，系统的组成结构如图 3-6 所示。

机架用来固定模块、提供模块工作电压和实现局部接地，并通过信号总线将不同模块连接在一起。S7-400 的模块插座焊在机架中的总线连接板上，模块插在模块插座上，有不同槽数的机架供用户选用，如果一个机架容纳不下所有的模块，可以增设一个或数个扩展机架，各机架之间用接口模块和通信电缆交换信息。

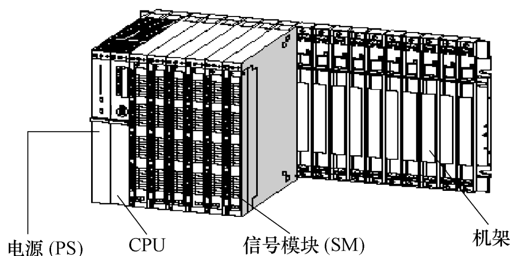


图 3-6 系统的组成结构

一、S7-400 的系统结构

S7-400 PLC 的背板总线是集成在背板之上的，这一点与 S7-300 不同。模块插在背板的插座上，有不同插槽数目的背板供用户选择。中央机架必须配置 CPU 和电源模块，电源模块应安装在机架的最左边。中央机架最多可插入 6 块发送型的接口模块，最多只有两个 IM 可以提供 5V 电源。通过 C 总线（通信总线）的数据交换仅限于 CC（中央控制器）和 6 个 EU（或称扩展单元）之间，最多能连接 21 个扩展机架。

中央机架必须配置 CPU 和一个电源模块，可以安装除用于接收的 IM（接口模块）外的所有 S7-400 模块。如果有扩展机架，中央机架和扩展机架都需要安装接口模块。

扩展机架可以安装除 CPU、发送 IM 、IM 463-2 适配器外的所有 S7-400 模块。但是电源模块不能与 IM 461-1（接收 IM）一起使用。关于扩展机架的详细配置与说明见第三章第五节 S7-300/400 扩展机架的配置与说明。

集中式扩展适用于小型配置或一个控制柜中的系统。CC 和 EU 的最大距离为 1.5m（带 5V 电源）或 3m（不带 5V 电源）。分布式扩展适用于分布范围广的场合，CC 与最后一个 EU 的最大距离为 100m（S7 EU）或 600m（S5 EU）。

用 E/T200 分布式 I/O 可以进行远程扩展，用于分布范围很广的系统。通过 CPU 中的 PROFIBUS-DP 接口，最多连接 125 个总线节点。使用光缆时 CC 和最后一个节点的距离为 23km。电源模块应安装在机架的最左边（第 1 槽），有冗余功能的电源模块是一个例外。中央机架只能插入最多 6 块发送型的接口模块，每个模块有两个接口，每个接口可以连接 4 个扩展机架，最多能连接 21 个扩展机架。中央机架中同时传送电源的发送接口模块（IM 460-1）不能超过两块，IM 460-1 的每个接口只能带一个扩展机架。扩展机架中的接口模块只能安装在最右边的槽（第 18 槽或第 9 槽）。通信处理器 CP 只能安装在编号不大于 6 的扩展机架中。

电源模板：将 SIMATIC S7-400 连接到 120/230 VAC 或 DC24V 电源上。

中央处理单元（CPU）：有多种 CPU 可供用户选择，有些带有内置的 PROFIBUS-DP 接口，用于各种性能范围。一个中央控制器可包括多个 CPU，以增强其性能。

各种信号模板（SM）：用于数字量输入和输出。

通信模板（CP）：用于总线连接和点到点的连接。

功能模板（FM）：专门用于计数、定位、凸轮控制等任务。

根据用户需要还提供以下部件：接口模板（IM），用于连接中央控制单元和扩展单元。

二、S7-400 的优点

1）运行速度快，CPU 417-4 的浮点数乘法运算只需要 0.09μs。

2）存储器容量大，例如 CPU 417-4 的工作存储器为 20MB，装载存储器可扩展至 64MB。

3）I/O 扩展功能强，可扩展 21 个扩展机架，CPU 417-4 最多可扩展 262144 个数字量 I/O 点和 16384 个模拟量 I/O 点。

4）有极强的通信能力，大多数 CPU 集成有 PROFIBUS-DP 接口，容易实现高速的分布式系统。

5）通过钥匙开关和口令实现安全保护。

6）诊断功能强。

7）集成的 HMI 服务，用户只需要为 HMI 服务定义源和目的地址，系统会自动传送信息。

三、S7-400 的通信功能

S7-400 有很强的通信功能，CPU 集成有 MPI 和 DP 通信接口，有 PROFIBUS-DP 和工业以太网通信模块，以及点到点通信模块。通过 PROFIBUS-DP 或 AS-i 现场总线，可以周期性地自动交换 I/O 模块的数据。在自动化系统之间，PLC 与计算机和 HMI 站之间，均可以交换数据。数据通信可以周期性地自动进行或基于事件驱动，由用户程序调用。

S7/C7 通信对象的通信服务通过集成在系统中的功能块来进行。可提供的通信服务有：使用 MPI 的标准 S7 通信；使用 MPI、C 总线（通信总线，Communication bus）、PROFIBUS-DP 和工业以太网的 S7 通信；与 S5 通信对象和第三方设备的通信。这些服务包括通过 PROFIBUS-DP 和工业以太网的 S5 兼容通信和标准通信。

第四节　机架与接口模块

一、机架

S7-400 的模块是用机架上的总线连接起来的。机架上的 P 总线（I/O 总线）用于 I/O 信号的高速交换和对信号模块数据的高速访问。C 总线（通信总线，或称 K 总线）用于在 C 总线各站之间的高速数据交换，两种总线分开后，控制和通信分别有各自的数据通道，通信任务不会影响控制的快速性。

1. 通用机架 UR1/UR2

UR1（18 槽）和 UR2（9 槽）有 P 总线和 K 总线，可以用作中央机架和扩展机架。它们用作中央机架时，可以安装除接收 IM 外的所有 S7-400 模块。

2. 中央机架 CR2/CR3

CR2 是 18 槽的中央机架，P 总线分为两个本地总线段，分别有 10 个插槽和 8 个插槽。两个总线段都可以对 K 总线进行访问。CR2 需要一个电源模块和两个 CPU，每个 CPU 有它自己的 I/O 模块，它们能相互操作和并行运行。CR3 是 4 槽的中央机架，有 I/O 总线和通信总线。

3. 扩展机架 ER1 和 ER2

ER1 和 ER2 是扩展机架，分别有 18 槽和 9 槽，只有 I/O 总线，未提供中断线，没有给模块供电的 24V 电源，可以使用电源模块、接收 IM 模块和信号模块。但是电源模块不能与 IM461-1 接收 IM 一起使用。

4. UR2-H 机架

UR2-H 机架用于在一个机架上配置一个完整的 S7-400H 冗余系统，也可以用于配置两个具有电气隔离的独立运行的 S7-400 CPU，每个均有自己的 I/O。UR2-H 需要两个电源模块和两个冗余 CPU。

二、接口模块

IM460-X 是用于中央机架 UR1、UR2 和 CR2 的发送接口模块；IM 461-X 是用于扩展机架 UR1、UR2 和 ER1、ER2 的接收接口模块。

1）IM460-0 和 IM461-0 分别是配合使用的发送接口模块和接收接口模块，属于集中式扩展，最大距离为 3m。IM460-0 有两个接口，每个接口最多扩展 4 个机架，模块最多可扩展 8 个机架，中央机架可以插 6 块 IM461-3 。

2）IM 460-1 和 IM 461-1 分别是配合使用的发送接口模块和接收接口模块，属于集中式扩展，最大距离为 1.5m 。中央控制器通过接口模块给扩展机架提供 5V 电源，最多能连接两个扩展机架，每个接口 1 个。

3）IM 460-3 和 IM 461-3 分别是配合使用的发送接口模块和接收接口模块，属于分布式扩展，最大距离为 100m，传输 C 总线和 P 总线。IM 460-3 有两个接口，每个接口最多扩展 4 个机架，模块最多扩展 8 个机架，中央机架可以插 6 块 IM 461-3。

4）IM 460-4 和 IM 461-4 分别是发送接口模块和接收接口模块，它们必须配合使用，属于分布式扩展，最大距离为 605m ，通过 P 总线传输数据。IM 460-4 有两个接口，每个接口最多扩展 4 个机架，模块最多可扩展 8 个机架，中央机架可以插 6 块 IM 461-4。

5）IM463-2 是发送接口模块，用于 S5 扩展机架的分布式扩展，最大距离为 600m，有两个接口，最多可扩展 8 个 S5 扩展机架，每个接口最多扩展 4 个机架，只能与 IM 314 配合使用。中央机架最多插 4 块 IM 463-2 。

6）IM 467 和 IM 467 FO 将 S7-400 作为主站接入 PROFIBUS-DP 网络，可以将多达 14 条 DP 线连接到 S7-400，IM 467 FO 集成了光纤接口。它们提供 PROFIBUS-DP 通信服务和 PG/OP 通信，以及通过 PROFIBUS-DP 的编程和组态，支持 SYNC/FREEZE 等距离和站点间通信功能。

三、错误诊断

1. 模板的诊断和过程监视

SIMATIC S7-400 的许多 I/O 模板具有智能性，可以用诊断功能来确定模板信号获取（就数字量模板而言）或模拟量信号处理（模拟量模板）是否正确地按要求进行。在诊断评价中，可参数化和不可参数化的诊断信息是有区别的。

可参数化的诊断信息：一个诊断信息只有通过相关参数化功能才能发出。

不可参数化的诊断信息：这些信息在任何情况下都能发出而不依赖于参数化功能。

如果一个诊断信息正在进行之中（例如“不提供编码”），模板起动一个诊断中断（在可参数化诊断信息的情况下是在相关参数化功能完成之后）。CPU 中断用户程序以及低优先级的程序的执行，而处理相关的诊断中断块（OB 82），根据模板的类型不同，有各种各样的诊断信息，具体见表 3-4、表 3-5 和表 3-6。

表 3-4 数字量 I/O 错误诊断表

诊断信息	可能发生故障/错误的原因	诊断信息	可能发生故障/错误的原因
没提供编码器	编码器超载,编码器到 M 的线路短路	无内部附加电压	模板没有给 L+电压
无外部附加电压	模板没有 L+电压	熔丝烧断	内部熔丝损坏
模板中的参数不对	不正确的参数传送到了模板	RAM 故障	周期性高电磁干扰模板损坏
监视器断路	周期性的高电磁干扰模板损坏	硬件中断丢失	硬件中断序列的到达速度快于 CPU 能够处理的能力
EPROM 故障	周期性的高电磁干扰模板损坏	—	—

表 3-5　模拟量输出模板错误诊断表

诊断信息	可能发生故障/错误的原因	诊断信息	可能发生故障/错误的原因
无外部负载电压	模板无 L+负载电压	对地短路	输出超负载，从 QV 到 MANA 的输出短路
配置/参数化错误	不正确的参数传送到了模板	线路中断	执行器电阻太高，模板和执行器之间的线路中断，回路未使用(开路)

表 3-6　模拟量输入模板错误诊断表

诊断信息	可能发生故障/错误的原因	诊断信息	可能发生故障/错误的原因
无外部负载电压	模板无 L+负载电压	测量范围下溢	输入值超出低限范围，引起故障的原因可能是： 测量值范围：4～20mA，1～5V 1）传感器连接极性颠倒 2）选择的测量范围有误 3）其他值的测量范围
配置/参数化错误	不正确的参数传送到了模板		
共模故障	输入(M)和测量回路(MANA)的参考势点的势差太高	超出测量信号上限范围	选择的测量范围有误，输入超高限
线路中断	编码器的连接电阻太高，模板和传感器之间的线路中断，回路没有连上(开路)		

2. 硬件中断

硬件中断功能用来监视过程信号以及反应信号变化的断开信号。

（1）数字量输入模板

依据参数化功能，模板可以在任选的每一个通道组的信号上升沿、下降沿或者一个信号变化时的两种跳变沿的任一个上初始化一个过程中断。CPU 中断用户程序或者具有低优先级的程序的执行，而处理相关的诊断中断块（OB40）。信号模板的每个通道可暂时存贮一个中断内容。

（2）模拟量输入模板

由参数化的上限值和下限值决定模拟量输入值的工作范围。此模板用这些极限值与数字化了的测量值相比较，如果测量值超过了其中一个极限值，则给出一个硬件中断。CPU 中断用户程序或具有较低优先级的功能块的执行，去处理相关的诊断中断功能块，如果这些极限值不在测量值范围之内，则不进行比较工作。

四、冗余设计

在许多自动化领域中，要求容错和高可靠性的自动化系统的应用越来越多。特别是在某些领域，停机将带来巨大的经济损失。在这种情况下，只有冗余系统才能满足高可靠性的要求。高可靠性的 SIMATIC S7-400H 能充分满足这些要求。它能连续运行，即使控制器的某些部件由于一个或几个故障而失效也不受影响。由于 SIMATIC S7-400H 具有很高的可用性，它特别适合于以下的应用领域：

1）控制器发生故障后再起动的费用十分昂贵（一般在过程控制工业中）。

2）如发生停机，将会造成重大的经济损失。

3）过程控制中包含有贵重的材料（如制药工业）。

4）无人管理的应用场合。

5）需减少维护人员的场合。

S7-400H 按冗余方式设计，CPU、电源模块是双重的，可以在故障发生后继续使用备用部件。此外，用户也可以自行决定其他需要双重的部件以增强设备的冗余性。S7-400H 的双重部件采用“热备用”模式的主动冗余原理，两个子单元都处于运行状态，采用“事件驱动同步”，当故障发生时，保证在双重部件之间无扰动切换。CPU 417-H 的操作系统自动地执行 S7-400H 需要的附加功能，包括数据通信，故障响应（切换到备用控制器），两个子单元的同步和自检功能等。冗余连接如图3-7所示。

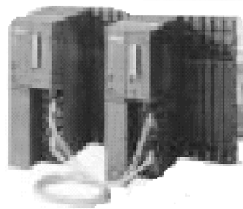


图 3-7 冗余连接图

为了保证无扰动切换，必须实现中央控制器链路之间快速、可靠的数据交换。两个控制器必须使用相同的用户程序，自动地接收相同的数据块、过程映像和相同的内部数据，例如定时器、计数器、位存储器等。这样可以确保两个子控制器同步地更新内容，在任意一个子系统有故障时，另一个子系统可以承担全部控制任务。

S7-400H 采用“事件驱动同步”，当两个子单元的内部状态不同时，例如在直接 I/O 访问、中断、报警和修改实时钟时，就会进行同步操作。通过通信功能修改数据，由操作系统自动执行同步功能，不需要用户编程。

S7-400H 对中央控制器之间的连接、CPU、处理器/ASIIC 和存储器进行自检。在起动后每个子单元完整地执行所有的测试功能。自检功能被分为几部分，每个周期只执行部分自检功能，以减轻 CPU 的负担。

第五节 S7-300/400 扩展机架的配置与说明

一、S7-300 系统扩展

通常一套 S7-300 系统有一个主机架，安装有 CPU 的机架称为主机架，当主机架上的 I/O 模块（最多 8 块）上的控制点数不够时，可以再增加 1～3 个扩展机架，每个扩展机架最多可安装 8 个 I/O 模块，装在 4～11 槽，3 个扩展机架最多安装 24 个 I/O 模块。

在使用扩展机架时，需要机架（Rack）、电源模块（PS）、接口模块（IM）、连接电缆 368、S7-300 的模块（信号模块、通信模块、功能模块等）。S7-300 的安装机架是一种导轨。你可以使用该导轨，安装 S7-300 系统的所有模板。S7-300 既可以水平安装，也可以垂直安装。要注意其允许的环境温度为：垂直安装：0～40℃；水平安装：0～60℃。CPU 和电源必须安装在左侧或底部。

应配合模板的安装宽度选择不同长度的导轨，不同模板的宽度可查样本得知，模拟 I/O 模板和数字 I/O 模板的宽度一般为 40mm。必须保持图 3-8 中所示的间隙，以提供模板安装空间，确保模板散热良好。

1. 使用单机架或多机架

是使用一个机架还是使用多个机架，取决于具体情况。

在下面的情况下应该使用单机架：结构紧凑、需要节约空间；CPU312、312IFM、312C 和 CPU313 只能用单机架；所需处理的信号量少。

在下面的情况下应该使用多机架：所需处理的信号量大；没有足够的插槽。如需将 S7-300 装在几个机架上，则需要接口模板（IM），接口模板的使命是将 S7-300 背板总线从一个机架扩展到下一个机架。中央处理单元 CPU 总是在 0 号机架上。接口模板又分如下两种，见表 3-7。

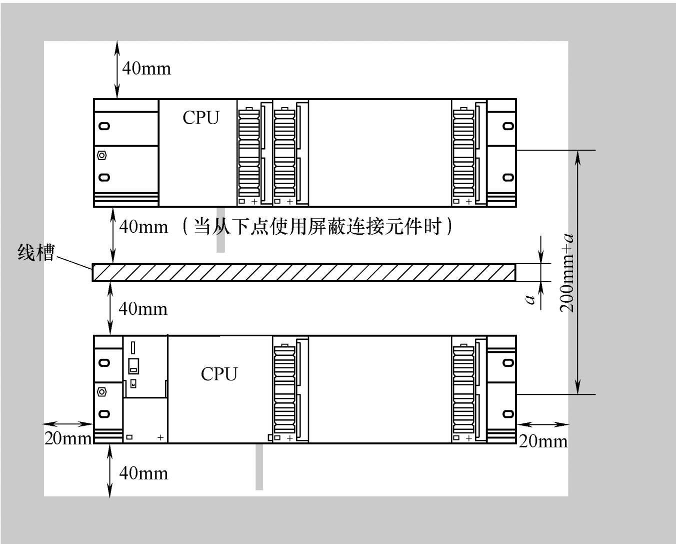


图 3-8　间隙

表 3-7　两种接口模板

特　　点	双线和多线配置	低成本双线配置
机架 0 中的发送接口模板	IM360 订货号:6ES7 360-3AA01-0AA0	IM365 订货号:6ES7 365-0AB00-0AA0(基本温度)6ES7 365-0BA81-0AA0(扩展温度)
机架 1 ~ 3 中的接收接口模板	IM361 订货号:6ES7 361-3CA01-0AA0 外接 DC24V 电源	IM365（硬连线至发送接口模板 IM365） 由发送 IM365 供电
扩展装置的最大数量	3	1
连接电缆长度	1m(6ES7368-3BB01-0AA0) 2. 5m(6ES7368-3BC51-0AA0) 5m(6ES7368-3BF01-0AA0) 10m(6ES7368-3CB01-0AA0)	1m(硬连线)
总线	P 总线(外设总线,I/O) C 总线(通信总线,也称 K 总线)	P 总线(外设总线 I/O) *

* IM365 扩展机架支持 P 总线，只能使用信号模板。当扩展机架使用 FM、CP 模块时，请选择 IM360/361 扩展模式。请参考以下网站：

http：//www4. ad. siemens. de/WW/view/en/19182754

http：//www4. ad. siemens. de/WW/view/en/188879

图 3-9 所示为一台 S7-300PLC 的模板在 4 个模板机架上的安装情况。

2. 主机架的配置方法

添加主机架如图 3-10 所示。

在 STEP7 中，通过简单地拖放操作就可以完成主机架的配置。在配置过程中，添加到主机架中模板的订货号（在硬件目录中选中一个模板，目录下方的窗口会显示该模板的订货号以及描述）应该与实际硬件一致。

首先直接新建一个项目，在项目中插入一个 SIMATIC 300 Station，双击 Hardware 图标，打开硬件组态程序。在硬件目录中找到 S7-300 机架，拖拽到左上方的视图中，即可添加一个主机架。

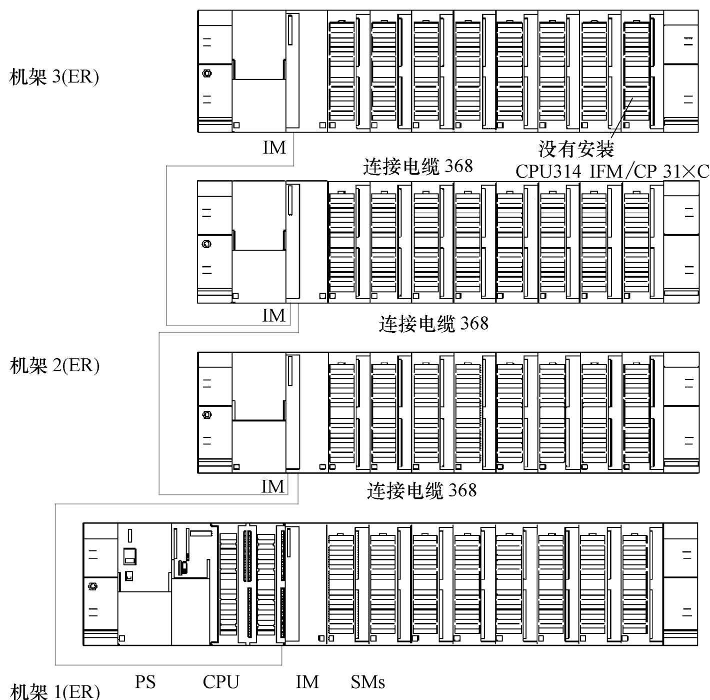


图 3-9 安装举例

1）插入主机架后，分别向机架中的 1 号槽添加电源，向 2 号槽添加 CPU。硬件目录中的某些 CPU 型号有多种操作系统版本，在添加 CPU 时，CPU 的型号和操作系统版本都要与实际硬件一致。

向主机架中添加电源和 CPU 如图 3-11 所示。

2）如果需要扩展机架，则应该在 IM-300 目录下找到相应的接口模板，添加到 3 号槽。如无扩展机架，3 号槽留空。

3）4～11 号槽中可以添加信号模板、功能模板、通信处理器等，上述模板分别在硬件目录中的 SM-300、FM-300 和 CP-300 目录下。

向主机架中添加信号模板、功能模板、通信处理器等如图 3-12 所示。

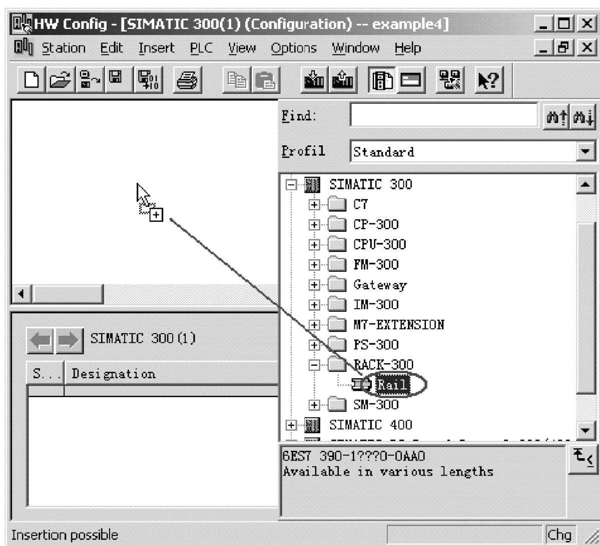


图 3-10 添加主机架

3. 机架扩展

一个 S7-300 站最多可以有一个主机架（0 号机架），3 个扩展机架（1～3 号机架）。主机架和扩展机架通过接口模板（IM）连接。

机架扩展有以下两种情况：

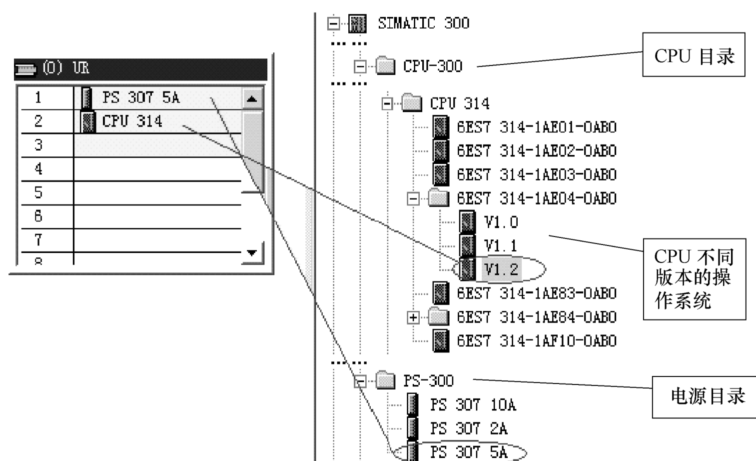


图 3-11　向主机架中添加电源和 CPU

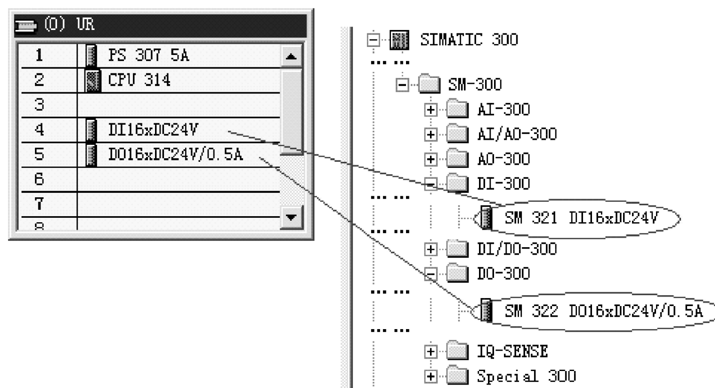


图 3-12　向主机架中添加信号模板、功能模板、通信处理器等

1）只有一个扩展机架时，主机架（0）和扩展机架（1）的 3 号槽中都使用 IM365 连接。

2）有 1～3 个扩展机架时，主机架（0）的 3 号槽中使用 IM360，扩展机架 1～3 的 3 号槽中用 IM361。

在 STEP7 中，可以像添加主机架一样，通过拖拽向站窗口中添加扩展机架。然后分别在主机架和扩展机架中添加相应的接口模板。STEP7 就会显示出相应的机架之间的连接。

机架扩展的示例如图 3-13 所示。

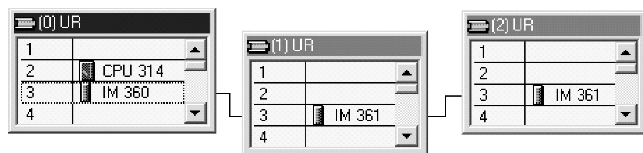


图 3-13　机架扩展示例

二、S7-400 系统扩展

中央机架可插入 4 个发送接口模块，最多可连 21 个扩展单元。

S7-400 机架扩展配置示例如图 3-14 所示。

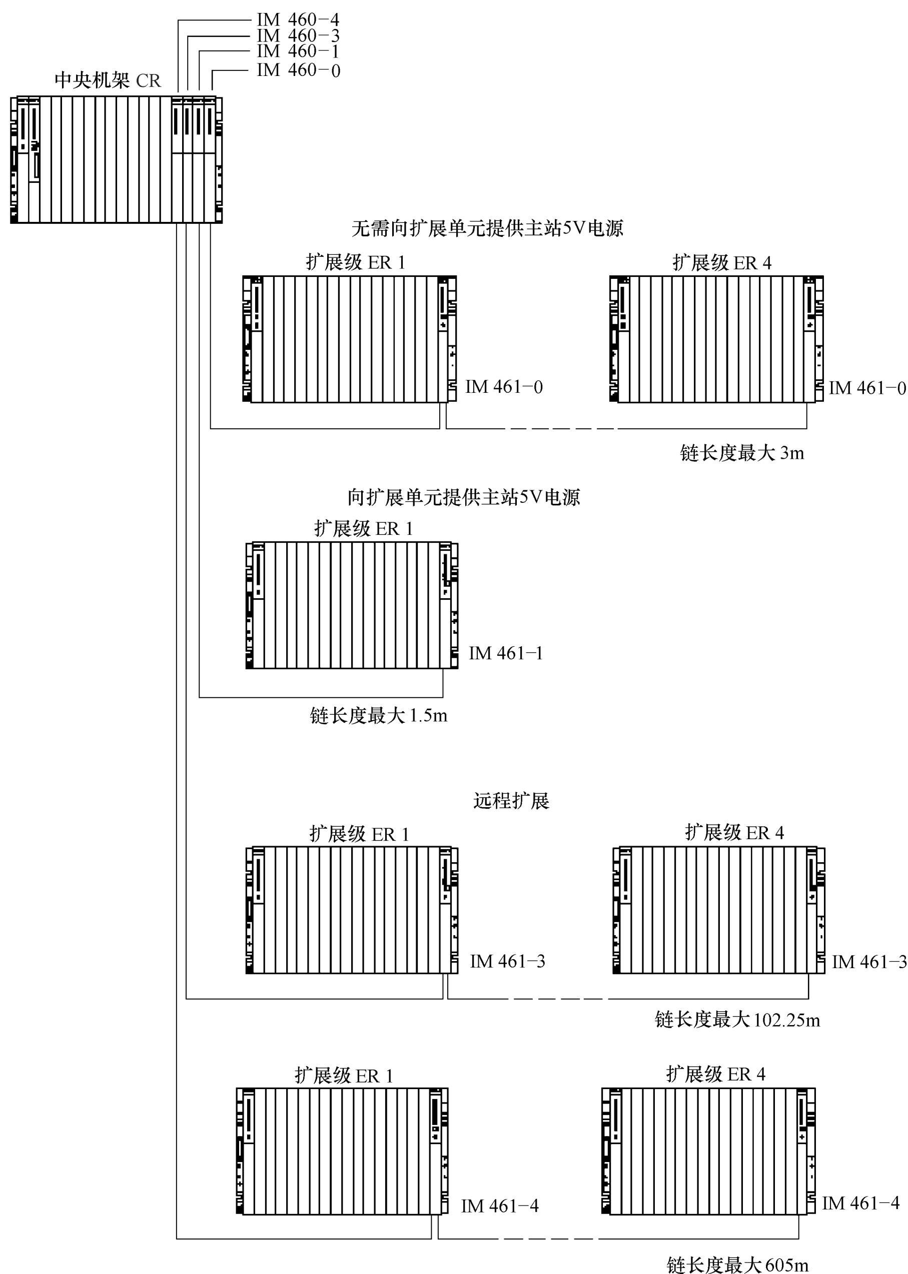


图 3-14　S7-400 机架扩展配置示例

1. IM460-0/461-0 模式

IM460-0：6ES7460-0AA00-0AB0：发送接口模块，扩展 8ER，Max. 3m。

6ES7460-0AA01-0AB0：发送接口模块，扩展 8ER，Max. 5m。

扩展机架需插电源模块，P 总线、C 总线（K 总线）。

IM461-0：6ES7461-0AA00-0AA0：接收接口模块。

6ES7461-0AA01-0AA0：接收接口模块。

终端电阻：6ES7461-0AA00-7AA0。

连接电缆：468-1。

S7-400IM 接口模块 460/1-0 面板图如图 3-15 所示。

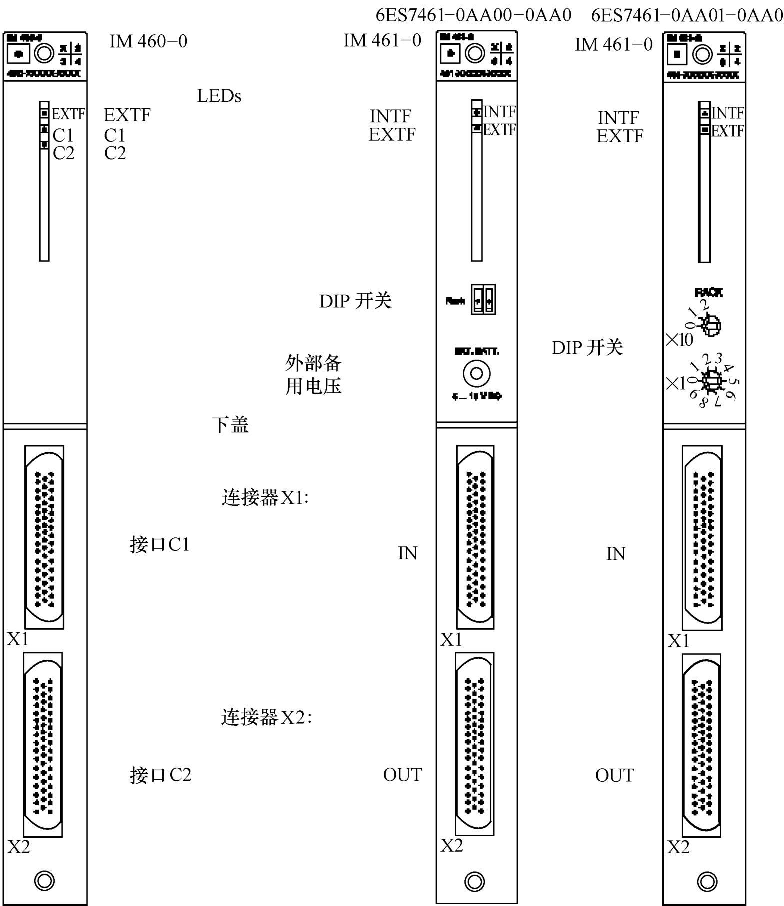


图 3-15　S7-400IM 接口模块 460/1-0 面板图

发送模块指示灯与状态见表 3-8。

表 3-8　发送模块指示灯与状态

EXTF LED（红灯）	当扩展 C1 或 C2 故障时 （没插终端电阻或电缆断）
C1 灯（绿灯）	扩展 C1 运行正常（连接端子 X1）
C1 灯（绿灯闪）	C1 连接的一个扩展单元没准备好 ● 没上电 ● 模块没有初始化
C2 灯（绿灯）	扩展 C2 运行正常（连接端子 X2）
C2 灯（绿灯闪）	C2 连接的一个扩展单元没准备好 ● 没上电 ● 模块没有初始化
连接端子 X1 和 X2	C1 和 C2 连接端子 X1 = 上部连接端子、X2 = 下部连接端子

接收模块指示灯与状态见表 3-9。

表 3-9 接收模块指示灯与状态

INTF LED(红灯)	当设置大于 21 或等于 0 的单元号,在电压低的情况下改变了单元号,红灯亮
EXTF LED(红灯)	外部故障时(电缆故障或模块没有初始化)
DIP 开关	设置扩展单元号
EXT. BATT 外部备用电压连接口	在 IM461-0 上(订货号 6ES7461-0AA00-0AA0)可接一个外部或中央备用电压(5～15V)(参考安装手册,第九章),以保证更换电源模块时,扩展单元正常运行。为满足柜内安装的空间要求,可使用带角度的电源连接线
连接端子 X1	上部连接端子(输入)连接上一个接口模块
连接端子 X2	下部连接端子(输出)连接下一个接口模块或终端器

2. IM460-1/IM461-1 模式

IM460-1：6ES7460-1BA00-0AB0：发送接口模块，扩展 2ER，Max. 1.5m。

6ES7460-1BA01-0AB0：发送接口模块，扩展 2ER，Max. 1.5m。

扩展机架无需电源模块，P 总线。当使用诊断功能、硬件中断、FMP 模块时，请选择其他扩展模式。

IM461-1：6ES7461-1BA00-0AA0：接收接口模块。

6ES7461-1BA01-0AA0：接收接口模块。

终端电阻：6ES7461-1BA00-7AA0。

连接电缆：468-3。

S7-400IM 接口模块 460/1-1 面板图如图 3-16 所示。

发送模块指示灯与状态见表 3-10。

表 3-10 发送模块指示灯与状态

EXTF LED(红灯)	当 C1 或 C2 所连接扩展故障时(没插终端电阻或电缆断)
C1 灯(绿灯)	扩展 C1 运行正常(连接端子 X1)
C1 灯(绿灯闪)	有没有初始化的模块
C2 灯(绿灯)	扩展 C2 运行正常(连接端子 X2)
C2 灯(绿灯闪)	有没有初始化的模块
连接端子 X1 和 X2	连接线 1 和线 2 的连接端子(出) X1 = 上部连接端子、X2 = 下部连接端子

接收模块指示灯与状态见表 3-11。

表 3-11 接收模块指示灯与状态

INTF LED(红灯)	当设置大于 21 或等于 0 的单元号,在电压低的情况下改变了单元号,红灯亮
EXTF LED(红灯)	外部故障时(电缆故障或模块没有初始化或中央单元断电)
5VDC(绿灯)	扩展单元电源正常
DIP 开关	设置扩展单元号
连接端子 X1	上部连接端子(输入)连接上一个接口模块
连接端子 X2	下部连接端子,旧模块(6ES7461-1BA00-0AA0)连接终端器(6ES7461-1BA00-7AA0);新模块(6ES7461-1BA01-0AA0)已集成终端器,无此端子

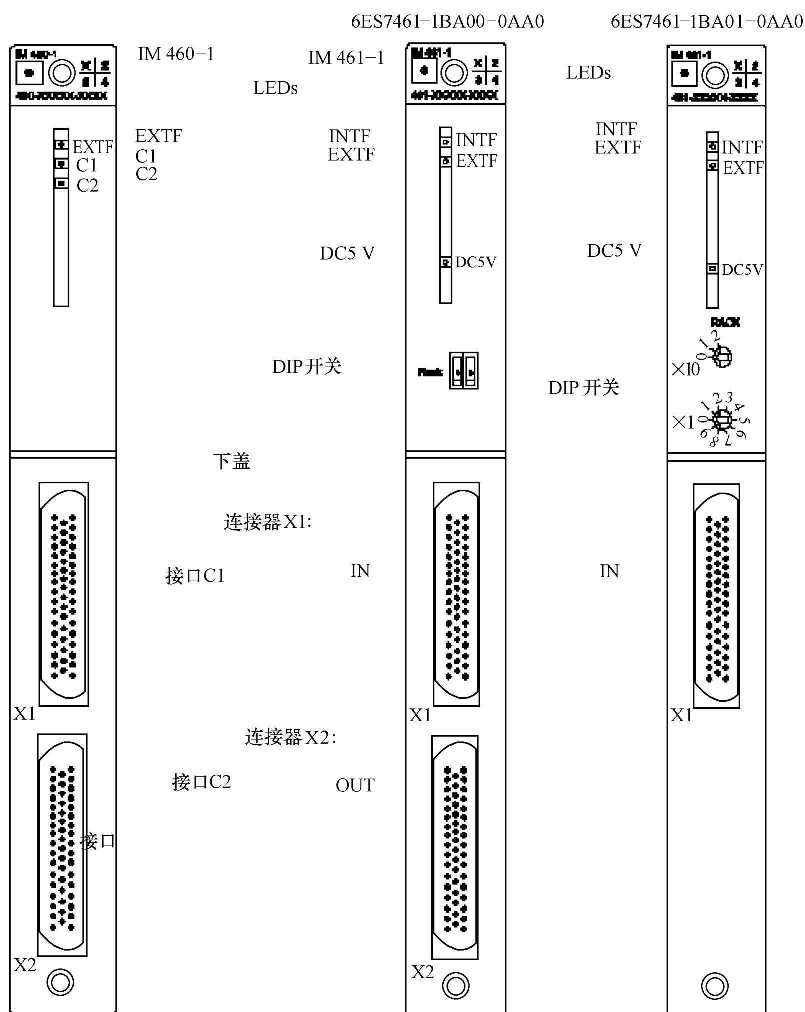


图 3-16　S7-400IM 接口模块 460/1-1 面板图

3. IM460-3/IM461-3 模式

IM460-3：6ES7460-3AA00-0AB0：发送接口模块，扩展 8ER，Max. 102m。

6ES7460-3AA01-0AB0：发送接口模块，扩展 8ER，Max. 102m。

扩展机架需插电源模块，P 总线、C 总线。

IM461-3：6ES7461-3AA00-0AA0：接收接口模块。

6ES7461-3AA01-0AA0：接收接口模块。

终端电阻：6ES7461-3AA00-7AA0。

连接电缆：468-1。

S7-400IM 接口模块 460/1-3 面板图如图 3-17 所示。

发送模块指示灯与状态见表 3-12。

接收模块指示灯与状态见表 3-13。

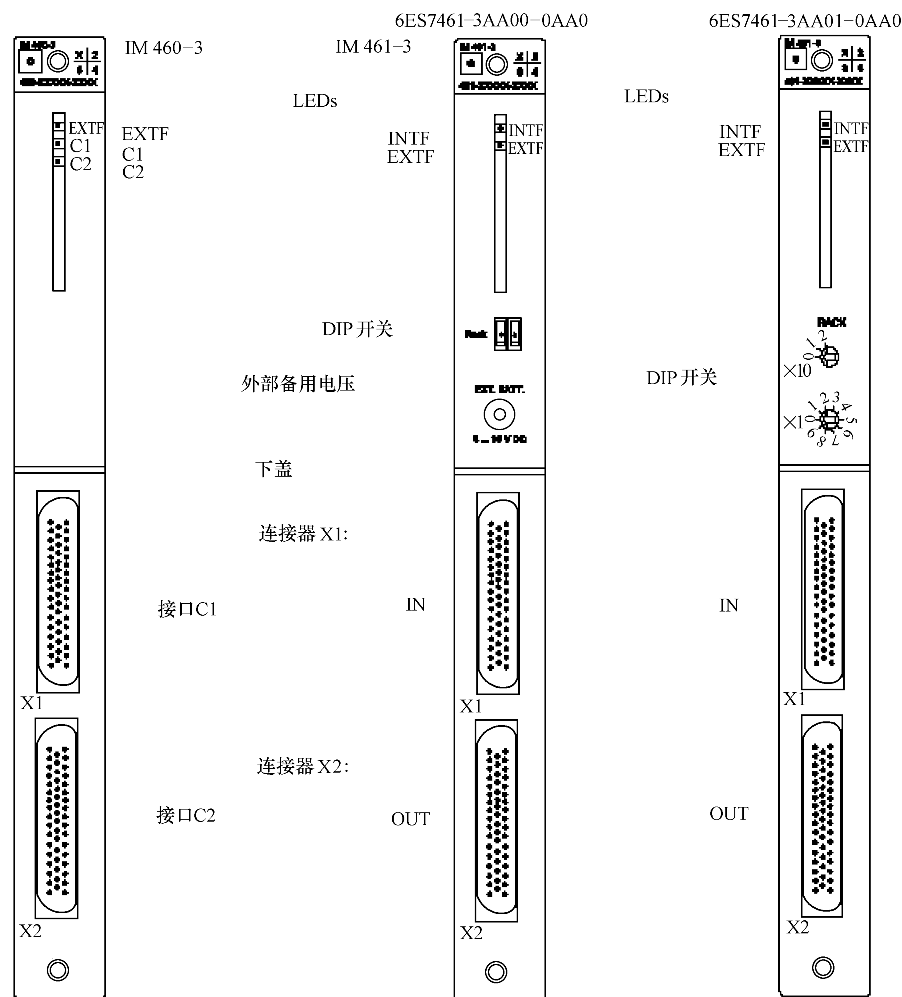


图 3-17　S7-400IM 接口模块 460/1-3 面板图

表 3-12　发送模块指示灯与状态

EXTF LED(红灯)	当 C1 或 C2 所连接扩展故障时 (没插终端电阻或电缆断)
C1 灯(绿灯)	扩展 C1(通过连接端子 X1)运行正常
C1 灯(绿灯闪)	C1 上有扩展单元没有准备好 • 电源模块没上电或 • 模块没有初始化
C2 灯(绿灯)	扩展 C2(通过连接端子 X2)运行正常
C2 灯(绿灯闪)	C2 上有扩展单元没有准备好 • 电源模块没上电或 • 模块没有初始化

表 3-13　接收模块指示灯与状态

INTF LED(红灯)	当设置大于 21 或等于 0 的单元号,在电压低的情况下改变了单元号,红灯亮
EXTF LED(红灯)	G 外部故障时(电缆故障或模块没有初始化或中央单元断电)
DIP 开关	设置扩展单元号

（续）

EX. BATT 外部备有电源插座	在 IM461-3（6ES7461-3AA00-0AA0）上可连接备用或中央电源（5 ~ 15V）以达到更换扩展单元电源模块时运行不中断。如果接受 IM 在柜内，为节省安装的空间，可使用带角度的电源连接线
连接端子 X1	上部连接端子（输入）连接上一个接口模块
连接端子 X2	下部连接端子连接下一个接口模块或终端器

4. IM460-4/IM461-4 模式

IM460-4：6ES7460-4AA00-0AB0：发送接口模块，扩展 8ER，Max. 605m。

6ES7460-4AA01-0AB0：发送接口模块，扩展 8ER，Max. 605m。

扩展机架需插电源模块，P 总线。

IM461-4：6ES7461-4AA00-0AA0：接收接口模块。

6ES7461-4AA01-0AA0：接收接口模块。

终端电阻：6ES7461-4AA00-7AA0。

连接电缆：468-1。

S7-400IM 接口模块 460/1-4 面板图如图 3-18 所示。

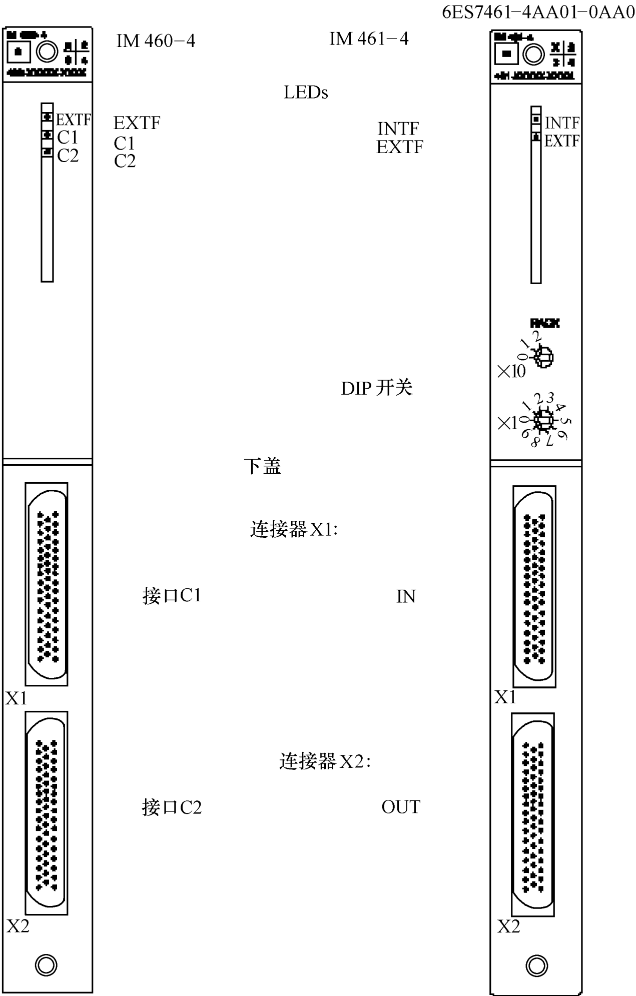


图 3-18　S7-400IM 接口模块 460/1-4 面板图

发送模块指示灯与状态见表 3-14。

表 3-14 发送模块指示灯与状态

EXTF LED(红灯)	当 C1 或 C2 所连接扩展故障时 (没插终端电阻或电缆断)
C1 灯(绿灯)	扩展 C1(通过连接端子 X1)运行正常
C1 灯(绿灯闪)	C1 上有扩展单元没有准备好 • 电源模块没上电或 • 模块没有初始化
C2 灯(绿灯)	扩展 C2(通过连接端子 X2)运行正常
C2 灯(绿灯闪)	C2 上有扩展单元没有准备好 • 电源模块没上电或 • 模块没有初始化

接收模块指示灯与状态见表 3-15。

表 3-15 接收模块指示灯与状态

INTF LED(红灯)	当设置大于 21 或等于 0 的单元号,在电压低的情况下改变了单元号,红灯亮
EXTF LED(红灯)	外部故障时(电缆故障或模块没有初始化或中央单元断电)
DIP 开关	设置扩展单元号
连接端子 X1	上部连接端子(输入)连接上一个接口模块
连接端子 X2	下部连接端子连接下一个接口模块或终端器

下列 CPU 不能使用 IM460-4 和 IM461-4 扩展模块:

- 6ES7412-1XF00-0AB0
- 6ES7413-1XG00-0AB0
- 6ES7413-2XG00-0AB0
- 6ES7414-1XG00-0AB0
- 6ES7414-2XG00-0AB0
- 6ES7416-1XJ00-0AB0

5. 用于连接 S5 模块的扩展机架

IM463-2:6ES7463-2AA00-0AA0 发送接口模块,装在 S7-400CR,传输距离 Max. 600m。扩展机架需电源,P 总线。

IM314:接收接口模块,装在 S5 扩展机架,可连接的机架有:

EU183U
EU185U
EU186U
ER701-2
ER701-3

终端电阻:6ES5760-1AA11。

如果您只是集中扩展,可使用以下接口模块:

IM300:6ES5300-5CA11
6ES5300-3AB11
6ES5300-5LB11

IM306:6ES5306-7LA11

接 S5 智能模板，使用 S5 适配器（S5ADAPTER）：6ES7 470-0AA00-0AA0

S7-400IM 接口模块 463-2 面板图如图 3-19 所示。

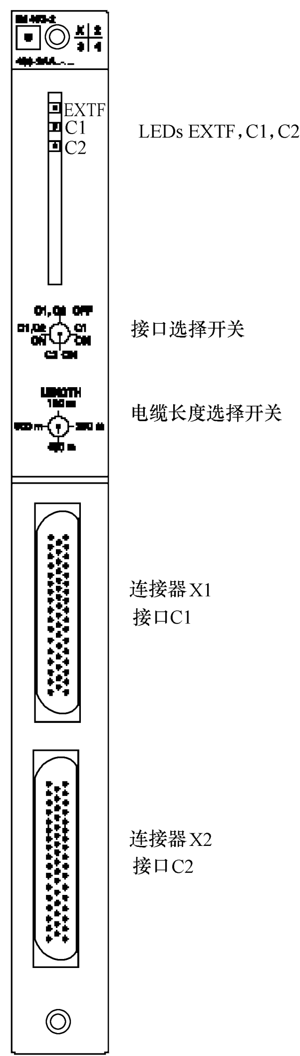


图 3-19　S7-400IM 接口模块 463-2 面板图

IM462-3 指示灯见表 3-16。

表 3-16　IM462-3 指示灯

指　示　灯	表　示　意　义
EXTF LED(红灯)	当 C1 或 C2 故障时红灯亮 (扩展单元电源断电；没插终端器；电缆线断开；或接口选择开关设置错误)
C1 灯(绿灯)	扩展 1(通过连接端子 X1)运行正常
C2 灯(绿灯)	扩展 2(通过连接端子 X2)运行正常
连接端子 X1 和 X2	连接插头(输出)到 C1 和 C2 X1 = 上部前连接器 X2 = 下部前连接器

接口选择开关见表 3-17。

表 3-17 接口选择开关

开关位置	表示意义	开关位置	表示意义
C1 ON	只使用接口 C1	C1,C2 ON	使用接口 C1 和接口 C2
C2 ON	只使用接口 C2	C1,C2 OFF	两个接口都不使用。 暂不使用 S5 扩展单元

电缆长度选择开关见表 3-18。

表 3-18 电缆长度选择开关

开关位置	表示意义	开关位置	表示意义
100	电缆长度 1 ~ 100m	450	电缆长度 250 ~ 450m
250	电缆长度 100 ~ 250m	600	电缆长度 450 ~ 600m

接口模板 IM314 面板图如图 3-20 所示。

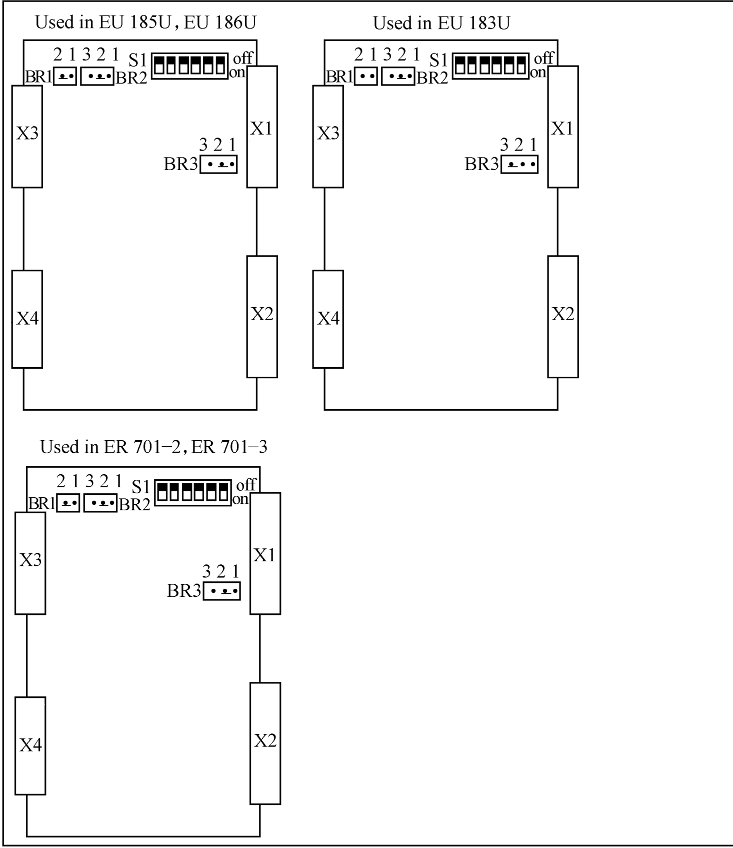


图 3-20 接口模板 IM314 面板图

I/O 地址区定义说明如图 3-21 所示。

721 电缆引脚定义如图 3-22 所示。

I/O 区域地址	选择位置
	0=OFF， 1=ON
P area： F000−F0FF	S1： 0000*)
Q area： F100−F1FF	0001
IM3 area：FC00−FCFF	1100
IM4 area：FD00−FDFF	1101

图 3-21 I/O 地址区定义说明

连接器 50 针接触	捆绑的 Ident.Sheath	识别薄片	核心颜色	连接器 50 针接触
20	1 No.16	红色	白色	20
21			棕色	21
4			绿色	4
5			黄色	5
18			灰色	18
19			粉色	19
2			蓝色	2
3			红色	3
24	2 No.17	绿色	白色	24
25			棕色	25
8			绿色	8
9			黄色	9
22			灰色	22
23			粉色	23
6			蓝色	6
7			红色	7
26	3 No.18	黄色	白色	26
27			棕色	27
10			绿色	10
11			黄色	11
42			灰色	42
43			灰色	43
44			蓝色	44
45			红色	45

图 3-22 721 电缆引脚定义

三、组态

S7-300：发送接口模块和接收接口模块都插 3 槽，接好电缆，上电，下载硬件组态即可。

S7-400：CR 机架最多 4 个发送 IM，无槽号限制。

ER 上接收 IM 插 18 槽（UR1、ER1）或 8 槽（UR2、ER2）。

双击发送 IM，通过 Connection 中的 Connect 将 EU 连接到 C1 或 C2。接好电缆，上电，下载硬件组态。

S7-400 扩展机架的连接如图 3-23 所示。

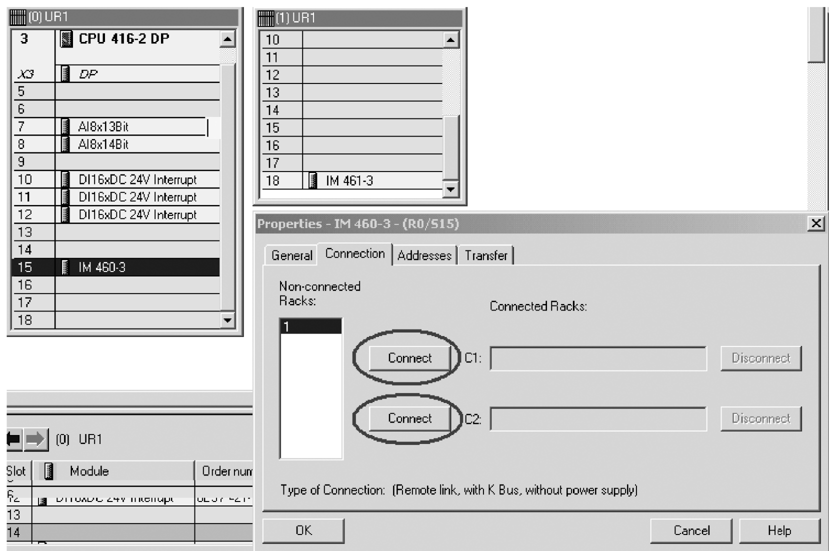


图 3-23 S7-400 扩展机架的连接

第六节 多 CPU 处理及 CPU 模块

一、多 CPU 处理

S7-400 提供了多种级别的 CPU 和种类齐全的具有通用功能的模块，使用户能根据需要组合成不同的专用系统。S7-400 采用模块化设计，性能范围不同的模块可以灵活组合，扩展十分方便。

1 个机架上最多可安装 4 个 CPU，并同时运行。这些 CPU 同时起动，同时进入 STOP 模式。多 CPU 处理适用的情况是：当用户程序过长，或者存储空间不够，需要分配给多个 CPU 来执行。可将系统分成不同的、相对独立的功能块，以利于彼此分开，单独控制，各 CPU 分别处理不同的部分，各自访问分配给自己的模块，给每个 CPU 分配模块的工作在 STEP 7 组态中进行。通过通信总线（C 总线），CPU 彼此互联，起动时，CPU 将自动检查彼此是否同步。

二、CPU 模块的元件

S7-400 有 7 种不同型号的 CPU，分别适用于不同等级的控制要求。不同型号的 CPU 面板上的元件不完全相同，CPU 内的元件封装在一个牢固而紧凑的塑料机壳内，面板上有状态和故障指示 LED、方式选择钥匙开关和通信接口。大多数 CPU 还有后备电池盒，存储器插槽可插入多达数兆字节的存储器卡。

CPU417 的工作存储器可以扩展，在 CPU 模块的存储器卡插槽内插入 RAM 存储卡，可以增加装载存储器的程序容量。Flash EPROM（快闪存储器）卡用来存储程序和数据，即使在没有后备电池的情况下，其内容也不会丢失。可以在编程器或 CPU 上编写 Flash 卡的内容，Flash 卡也可以扩展 CPU 装载存储区的容量。CPU417-4 和 CPU417-4H 还有存储器扩展接口，可以扩展工作存储器。集成式 RAM 不能扩展，集成装载存储器为 256KB（RAM），用存储器卡扩展 FEPROM

和 RAM 最大各 64MB。电池可以对所有的数据提供后备电源。

根据模块类型的不同，在 S7-400 的电源模块中可以使用一个或两个后备电池，为存储在内置的装载存储器和外部装载存储器、工作存储器的 RAM 中的用户程序和内部时钟提供后备电源，保持存储器中的存储器位、定时器、计数器、系统数据和数据块中的变量。

第四章 S7-300/400 PLC的常用指令

通过本章的学习主要掌握基本指令的格式、用法、注意事项和编程，了解 PLC 的编程语言及其特点，了解 PLC 的存储区、数据类型和寄存器功能。

第一节 S7-300/400 PLC 编程基础

一、编程语言

STEP 7 是西门子公司专为 S7-300/400 系列 PLC 设计的编程软件。标准的 STEP 7 软件包配备三种基本编程语言，即梯形图（LAD）、功能块图（FBD）和语句表（STL）编程语言，在 STEP 7 中可以相互转换；另外还有多种可选编程语言，如选用要收附加费。

1. 梯形图（LAD）编程语言

梯形图（LAD）是一种融逻辑操作、控制于一体，面向对象的、实时的、图形化的编程语言，类似继电器控制电路图。

梯形图由触点、线圈和指令框组成，直观易懂，有时也叫做电路或程序。触点代表输入，线圈代表运算结果，指令框用来表示计数器、定时器及运算等指令。梯形图按自上而下，从左到右的顺序排列，最左边的竖线称为起始母线也叫左母线，然后按一定的控制要求和规则连接各个接点，最后以继电器线圈（或再接右母线）结束，称为逻辑行或叫“梯级”。在 STEP 7 中把这样的由触点和线圈等组成的独立电路叫网络（Network）。通常一个梯形图中有若干逻辑行（梯级或网络），形似梯子，如图 4-1 所示，梯形图由此而得名。

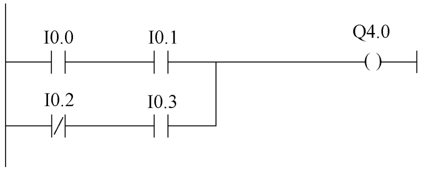


图 4-1 梯形图（LAD）

梯形图的逻辑运算，若没有跳转指令，在网络中按从左到右的方向执行，网络之间按从上到下的顺序执行，执行完所有的网络后，返回到最上面重新开始，循环执行。

梯形图信号流向清楚、简单、直观、易懂，很像电气原理图，易被电气工程人员理解和使用。梯形图在 PLC 中用的非常普遍，已成为 PLC 程序设计的基本语言。

2. 功能块图（FBD）编程语言

功能块图（FBD）是一种图形化的高级编程语言，对应于逻辑电路的图形语言。它类似于普通逻辑功能图，沿用了半导体逻辑电路的逻辑框图的表达方式。一般用一种功能方框表示一种特定的功能，框图内的符号表达了该功能块图的功能。

通过软连接的方式把所需的功能块图连接起来，用于实现系统的控制。其表达格式有利于程序流的跟踪。

功能块图有基本逻辑功能、计时和计数功能、运算和比较功能及数据传送功能等。

功能块图通常有若干个输入端和若干个输出端。输入端是功能块图的条件，输出端是功能块图的运算结果。

图 4-2 所示的功能块图（FBD），没有触点和线圈，也没有左、右母线的概念。信号自左向右流动。

3. 语句表（STL）编程语言

语句表（STL）是用助记符来表达 PLC 的各种控制功能的，它类似于计算机的汇编语言，但比汇编语言更直观易懂，编程简单，因此也是应用很广泛的一种编程语言。这种编程语言可使用简易编程器编程，但比较抽象，一般与梯形图语言配合使用，互为补充。目前，大多数 PLC 都有语句表编程功能，但各厂家生产的 PLC 语句表（STL）所用的助记符互不相同，不能兼容。图 4-3 所示的是语句表（STL）。

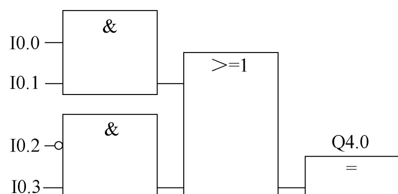


图 4-2　功能块图（FBD）

```
A    M0.0
AN   I0.0
=    Q4.0
```

图 4-3　语句表（STL）

通常梯形图（LAD）程序、功能块图（FBD）程序、语句表（STL）程序可有条件的方便地转换。但是，语句表（STL）可以编写梯形图（LAD）或功能块图（FBD）无法实现的程序。熟悉 PLC 和逻辑编程的有经验的程序员最适合使用语句表（STL）编程语言编程。

4. STEP 7 选用的编程语言和仿真软件

STEP 7 可选用的编程语言包括 S7-GRAPH、S7-SCL、S7-HiGRAPH 和 S7-CFC，下面分别给以介绍。

S7-GRAPH 用于编制顺序控制程序，特别适合于生产制造过程。

S7-SCL 是一种用于实现复杂的数学运算的高级文本语言，适合于熟悉高级编程语言的用户进行计算和数据处理。

S7-HiGRAPH 是用状态图（State Graphs）描述异步、非顺序过程的编程语言。

S7-CFC 是用图形连接程序库中各种以块形式提供的功能，从而实现编程的语言，适合于连续过程控制的编程。

S7-PLCSIM 仿真软件用于 SIMATIC S7 程序块的测试，能对 LAD、FBD、STL、S7-GRAPH、S7-HiGRAPH、S7-SCL 和 S7-CFC 编写的程序进行仿真。

二、数据类型

S7 的数据类型可分为三种：基本数据类型、复合数据类型和参数类型。

1. 基本数据类型

S7-300/400 PLC 的指令参数所用的基本数据类型有 1 位布尔型（BOOL）、8 位字节型（BYTE）、16 位无符号整数（WORD）、16 位有符号整数（INT）、32 位无符号双字整数（DWORD）、32 位有符号双字整数（DINT）、32 位实数型（REAL）、16 位 SIMATIC 时间（S5TIME）、32 位 IEC 时间（TIME）、16 位 IEC 日期（DATA）、32 位时间（TIME OF DAY）、8 位字符（CHAR）等。基本数据类型说明见表 4-1。

表 4-1　基本数据类型说明

数据类型	位数	格式选项	范　围
BOOL（位）	1	布尔文本	TRUE/FALSE
BYTE（字节）	8	十六进制的数字	B#16#0 ~ B#16#FF

（续）

数据类型	位数	格式选项	范围
WORD（字）	16	二进制的数字 十六进制的数字 BCD 十进制无符号数字	2#0～2#1111_1111_1111_1111 W#16#0～W#16#FFFF C#0～C#999 B#(0.0)～B#(255.255)
DWORD（双字）	32	二进制的数字 十六进制的数字 十进制无符号数字	2#0～ 2#1111_1111_1111_1111 1111_1111_1111_1111 DW#16#0000_0000～ DW#16#FFFF_FFFF B#(0,0,0,0)～ B#(255,255,255,255)
INT（整数）	16	十进制有符号数字	－32768～32767
DINT（整数，32 位）	32	十进制有符号数字	L#－2147483648～L#2147483647
REAL（浮点数）	32	IEEE 浮点数	上限:3.402823e＋38 下限:1.175 495e－38
S5TIME（SIMATIC 时间）	16	S7 时间 以步长 10ms(默认值)	S5T#0H_0M_0S_10MS～ S5T#2H_46M_30S_0MS 和 S5T#0H_0M_0S_0MS
TIME（IEC 时间）	32	IEC 时间步长为 1ms,有符号整数	-T#24D_20H_31M_23S_648MS～ T#24D_20H_31M_23S_647MS
DATE（IEC 日期）	16	IEC 日期步长为 1 天	D#1990-1-1～D#2168-12-31
TIME_OF_DAY（时间）	32	时间步长为 1ms	TOD#0:0:0.0～TOD#23:59:59.999
CHAR（字符）	8	ASCII 字符	‘A’,‘B’等

2. 复杂数据类型

复杂数据类型是指大于 32 位的数字数据群或包含其他数据类型的数据群，包括 DATE_ AND_ TIME 、STRING、ARRAY、STRUCT、UDT（用户自定义数据类型）、FB 和 SFB，见表 4-2。

表 4-2　复合数据类型说明

数据类型	说明
DATE_AND_TIME	日期-时间。定义具有 64 位(8 个字节)的区域。此数据类型以二进制编码的十进制的格式保存,存储年(字节 0)、月(字节 1)、日(字节 2)、小时(字节 3)、分钟(字节 4)、秒(字节 5)、毫秒(字节 6 和字节 7 的一半)和星期(字节 7 的另一半),用 BCD 码来表示。范围为 DT#1990-1-1-0:0:0.0～DT#2089-12-31-23:59:59.999
STRING	字符串。定义最多有 254 个字符的组(数据类型 CHAR)。为字符串保留的标准区域是 256 个字节长。这是保存 254 个字符和 2 个字节的标题所需要的空间。可以通过定义即将存储在字符串中的字符数目来减少字符串所需要的存储空间(例如:string[9]‘Siemens’)
ARRAY	数组。定义一个数据类型(基本或复杂)的多维组群。例如:“ARRAY [1..2,1..3] OF INT”定义 2×3 的整数数组。使用下标(“[2,2]”)访问数组中存储的数据。最多可以定义 6 维数组。下标可以是任何整数(－32768～32767)
STRUCT	结构。定义一个数据类型任意组合的组群,最多可以嵌套 8 层。例如,可以定义结构的数组或结构和数组的结构
UDT	用户数据类型。在创建数据块或在变量声明中声明变量时,简化大量数据的结构和数据类型的输入。在 STEP 7 中,可以组合复杂的和基本的数据类型以创建用户的“用户自定义”数据类型。UDT 具有自己的名称,因此可以多次使用
FB、SFB	块。确定分配的实例数据块的结构,并允许在一个实例数据块中传送数个 FB 调用的背景数据

3. 参数类型

参数类型是为块之间传送的形式参数而定义的数据类型。STEP 7 提供的参数类型有 TIMER 或 COUNTER（定时器和计数器）、BLOCK（块）、POINTER（指针）、ANY（任意参数）等，参数类型也可以在用户自定义数据类型（UDT）中使用。

定时器和计数器：赋值给 TIMER 或 COUNTER 参数类型的形参，相应的实际参数必须是定时器或计数器，换句话说，在正整数之后输入“T”或“C”。

块：指定用作输入或输出的特定块。参数的声明确定了使用的块类型（FB、FC、DB 等）。如果赋值给 BLOCK 参数类型的形参，指定块地址作为实际参数。实例：“FC101”（当使用绝对寻址时）或“Valve”（使用符号寻址）。

指针：参考变量的地址。指针包含地址而不是值。当赋值给 POINTER 参数类型的形参时，指定地址作为实际参数。在 STEP 7 中，可以用指针格式或简单地以地址指定指针（例如，M 50.0）。寻址以 M 50.0 开始的数据的指针格式的实例：P#M50.0

任意参数：当实际参数的数据类型未知或当可以使用任何数据类型时，可以使用该参数。

参数类型说明见表 4-3。

表 4-3　参数类型说明

参　　数	字节长度	描　　述
TIMER	2 个字节	指示程序在调用的逻辑块中使用的定时器 格式：T1
COUNTER	2 个字节	指示程序在调用的逻辑块中使用的计数器 格式：C10
BLOCK_FB BLOCK_FC BLOCK_DB BLOCK_SDB	2 个字节	指示程序在调用的逻辑块中使用的块 格式：FC101 DB42
POINTER	6 个字节	识别地址。 格式：P#M50.0
ANY	10 个字节	在当前参数的数据类型未知时使用 格式： P#M50.0 BYTE 10 数据类型的 ANY 格式 P#M100.0 WORD 5 L#1COUNTER 10 用于参数类型的 ANY 格式

三、存储器区域

CPU 存储器中存放的数据类型有 BOOL、BYTE、WORD、INT、DWORD、DINT、REAL。不同的数据类型具有不同的数据长度和数值范围。在上述数据类型中，用字节（B）型、字（W）型、双字（D）型分别表示 8 位、16 位、32 位数据的数据长度。不同的数据长度对应的数值范围见表 4-4。例如，数据长为字（W）型的无符号整数（WORD）的数值范围为 0 ~ 65535。

表 4-4　数据长度与数值

数据长度	无符号数		有符号数	
	十进制	十六进制	十进制	十六进制
B(字节型) 8 位值	0 ~ 255	0 ~ FF		
W(字型) 16 位值	0 ~ 65535	0 ~ FFFF	－32768 ~ 32767	8000 ~ 7FFF

（续）

数据长度	无符号数		有符号数	
	十进制	十六进制	十进制	十六进制
D(双字型) 32 位值	0 ~ 4294967295	0 ~ FFFF FFFF	− 2147483648 ~ 21474 83647	8000 0000 ~ 7FFF FFFF
R(实数型) 32 位值	$-10^{38} \sim +10^{38}$			

PLC 的存储器分为装载存储器、系统存储器、主存储器（也叫工作存储器）3 个区。

装载存储器区用于保存代码块和数据块以及系统数据（组态、连接和模块参数等），但标记有与运行时间无关的块不能保存在装载存储器中，有的可用微存储卡（MMC）扩展。

工作存储器用于存放与运行相关的块（代码块和数据块），其容量基本由所选的 CPU 决定，CPU 417 的工作存储器可以扩展。

系统存储器是每个 CPU 均可使用的存储器部件，包括存储位、定时器和计数器的地址区，I/O 过程映像，局域数据，块栈和中断栈。系统存储器是用户程序执行过程中的内部工作区域，存储器为 RAM。

1. 系统存储器存储区的地址表示格式

存储器是由许多存储单元组成，每个存储单元都有唯一的地址，可以依据存储器地址来存取数据。系统存储区地址的表示格式有位、字节、字、双字地址格式。

（1）位地址格式

系统存储器的存储区域某一位的地址格式为：Ax. y。

必须指定存储器区域标识符 A、字节地址 x 及位号 y。例如 I4. 5 表示图 4-4 中黑色标记的位地址，I 是变量存储器的区域标识符，4 是字节地址，5 是位号，在字节地址 4 与位号 5 之间用点号“.”隔开。图 4-4 中 MSB 表示最高位，LSB 表示最低位。

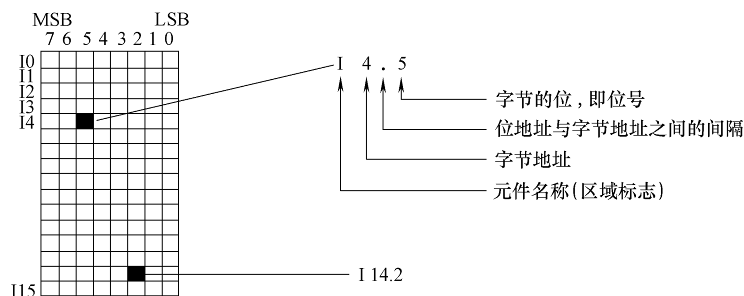


图 4-4　位地址格式

（2）字节、字、双字地址格式

系统存储器区域的字节、字、双字地址格式为：ATx。

必须指定区域标识符 A、数据长度 T 以及该字节、字或双字的起始字节地址。

图 4-5 中，用 MB100、MW100、MD100 分别表示字节、字、双字的地址。MW100 由 MB100、MB101 两个字节组成；MD100 由 MB100 ~ MB103 4 个组成。

（3）其他地址格式

系统存储器的存储区域中，还包括定时器（T）存储区、计数器（C）存储区等，它们的地址格式为：Ay，由区域标识符 A 和元件号 y 组成，例如 T32 表示某定时器的地址，T 是定时器的

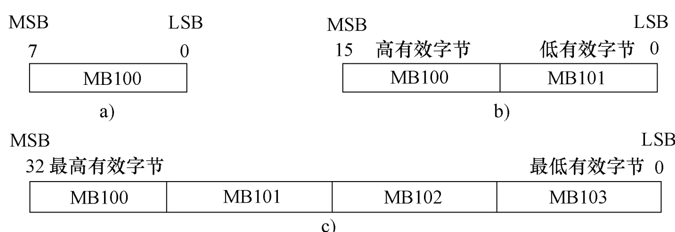


图 4-5　字节、字和双字

区域标识符，32 是定时器号，同时 T32 又可表示此定时器的当前值。

2. 系统存储器存储区域

（1）输入/输出映像寄存器（I/Q）

1）输入映像寄存器（I）。PLC 的输入端子是从外部接收输入信号的窗口。每一个输入端子与输入映像寄存器（I）的相应位相对应。输入点的状态，在每次扫描周期开始（或结束）时进行采样，并将采样值存于输入映像寄存器，作为程序处理时输入点状态的依据。输入映像寄存器的状态只能由外部输入信号驱动，而不能在内部由程序指令来改变。输入映像寄存器（I）的地址格式为：

位地址：I［字节地址］.［位地址］，如 I0.1。

字节、字、双字地址：I［数据长度］［起始字节地址］，如 IB4、IW6、ID10。

2）输出映像寄存器（Q）。每一个输出模块的端子与输出映像寄存器的相应位相对应。CPU 将输出判断结果存放在输出映像寄存器中，在扫描周期的结尾，CPU 以批处理方式将输出映像寄存器的数值复制到相应的输出端子上。通过输出模块将输出信号传送给外部负载。可见，PLC 的输出端子是 PLC 向外部负载发出控制命令的窗口。输出映像寄存器（Q）的地址格式为：

位地址：Q［字节地址］.［位地址］，如 Q4.1。

字节、字、双字地址：Q［数据长度］［起始字节地址］，如 QB100、QW11、QD10。

I/O 映像区实际上就是外部 I/O 设备状态的映像区，PLC 通过 I/O 映像区的各个位与外部物理设备建立联系，每个位都可以映像输入、输出单元上的每个端子的状态。

在程序的执行过程中，对于输入或输出的存取通常是通过映像寄存器，而不是实际的输入、输出端子。梯形图中的输入继电器、输出继电器的状态是对应于 I/O 映像寄存器相应位的状态。使得系统在程序执行期间完全与外界隔开，从而提高了系统的抗干扰能力。建立了 I/O 映像区，用户程序存取映像寄存器中的数据要比存取输入、输出物理点要快得多，因此加速了运算速度。此外，外部输入点的存取只能按位进行，而 I/O 映像寄存器的存取可按位、字节、字、双字进行，因而使操作更加灵活。

（2）内部标志位存储器（M）

内部标志位存储器（M）也称内部线圈，是模拟继电器控制系统中的中间继电器，它存放中间操作状态，或存储其他相关的数据。内部标志位存储器（M）以位为单位使用，也可以字节、字、双字为单位使用，其地址格式为：

位地址：M［字节地址］.［位地址］，如 M26.7。

字节、字、双字地址：M［数据长度］［起始字节地址］，如 MB11、MW23、MD26。

（3）外设输入（PI）和外设输出（PQ）

外设输入（PI）和外设输出（PQ）区允许直接访问本地的和分布式的输入模块和输出模块。

其存取格式为：

字节、字、双字地址：PI 或 PQ［数据长度］［起始字节地址］，如 PIB10、PIW100、PID120。

(4) 局部数据区（L）

局部数据区用来存放处理组织块、功能块和系统块时相应块的临时数据，可以按位、字节、字、双字访问局部数据区。局部数据区（L）的地址格式为：

位地址：L［字节地址］.［位地址］，如 L0.0。

字节、字、双字地址：L［数据长度］［起始字节地址］，如 LB33、LW44、LD55。

(5) 共享数据块（DB）和背景数据块（DI）

共享数据块（DB）供所有的逻辑块使用，而背景数据块（DI）与系统功能块和某一功能块相关联。用“OPN DB（或 DI）”指令打开。地址格式为：

位地址：DB 或 DI［字节地址］.［位地址］，如 DB3.1，DI3.3。

字节、字、双字地址：DB 或 DI［数据长度］［起始字节地址］，如 DBB4、DBW10、DID21。

(6) 定时器存储器（T）

定时器是模拟继电器控制系统中的时间继电器，时间值可用 BCD 码或二进制方式读取。定时器存储器地址表示格式为：T［定时器号］，如 T24。

(7) 计数器存储器（C）

计数器是累计其计数输入端脉冲电平由低到高的次数，有三种类型：增计数、减计数、增减计数。计数值可用 BCD 码或二进制方式读取，计数值的范围为 0～999。

其地址表示格式为：C［计数器号］，如 C5。

四、寻址方式

执行任何一条指令都需要操作数。寻址方式就是指令中用于说明操作数所在地址的方法，操作数可直接或间接给出。

S7-300/400 PLC 的寻址方式有：立即寻址、直接寻址、间接寻址，其中间接寻址又可分为存储器间接寻址和寄存器间接寻址。

1. 立即寻址

立即寻址方式是在指令中直接给出操作数，出现在指令中的操作数叫立即数，所以称为立即操作数或立即寻址。立即寻址方式可用来提供常数，设置初始值等。指令中经常使用常数。常数值可分为字节、字、双字型等数据。CPU 以二进制方式存储所有常数。指令中可用十进制、十六进制、ASCII 码或浮点数形式来表示。十进制、十六进制、ASCII 码浮点数的表示，见表 4-5。

表 4-5　十进制、十六进制、ASCII 码浮点数的表示

符　号	说　明	符　号	说　明
B#16#, W#16#, DW#16#	十六进制字节、字和双字常数	T#	IEC 时间常数
D#	IEC 日期常数	TOD#	实时时间常数(16/32 位)
L#	32 位整数常数	C#	计数器常数(BCD 编码)
P#	地址指针常数	2#	二进制常数
S5T#	S5 时间常数(16 位)	B(b1, b2) B(b1, b2, b3, b4)	常数 2 或 4 个字节

2#用来表示二进制常数，例如 2#0101 1110，#为常数的进制格式说明符。如果常数无任何格式说明符，系统默认为十进制。

立即寻址举例如下：

```
SET                          //将 RLO 置 1
L   +25                      //将 16 位整数常数“25”装入 ACCU1 中
L   L# -1                    //将 32 位整数常数“-1”装入 ACCU1 中
L   2#1000_ 1000_ 1000_ 1000 //将 16 位二进制常数装入 ACCU1 中
L   DW#16#A0E0_ ABCE         //将十六进制双字常数装入 ACCU1 中
L   “END”                    //将 ASCII 字符数装入 ACCU1 中
L   T#400MS                  //将时间值 400ms 装入 ACCU1 中
L   C#200                    //将计数值装入 ACCU1 中
L   B#（100，19）             //将 2 字节无符号常数 100 和 19 装入 ACCU1
L   B#（100，12，40，9）       //将 4 字节无符号常数装入 ACCU1
L   P# 10.1                  //将内部区域指针装入 ACCU1 中
L   P#Q20.5                  //将交叉区域指针装入 ACCU1 中
L   -1.6                     //将实数（浮点数）装入 ACCU1 中
L   D#2007_ 02_ 4            //将日期装入 ACCU1 中
L   TOD#20：20：20.200        //将实时时间装入 ACCU1 中
```

2. 直接寻址

直接寻址方式是，指令直接使用存储器或寄存器的元件名称和地址编号，根据这个地址就可以立即找到该数据。操作数的地址应按规定的格式表示。指令中，数据类型应与指令标识符相匹配。

不同数据长度的寻址指令举例如下：

```
位寻址：   A   I0.3     //输入位 I0.3 的“与”操作
字节寻址：L   IB5      //将输入字节 IB5 装入 ACCU1 的低字节
字寻址：   L   IW4      //将输入字 IW4 装入 ACCU1 的低字
双字寻址：L   ID20     //将输入双字 ID20 装入 ACCU1
```

3. 间接寻址

间接寻址方式是指在指令中到指定的存储单元地址（也称地址指针）取操作数地址。寻址中要用到寄存器，常用的寄存器有累加器、状态字寄存器、地址寄存器。间接寻址包括存储器间接寻址和寄存器间接寻址。

（1）累加器（ACCUx）

S7-300/400 的累加器（ACCUx）均为 32 位的累加器，用于处理字节、字或双字的寄存器。送入累加器的操作数在累加器中运算处理，并可保存在累加器中，处理的 8 位或 16 位数据结果放在累加器的低端。S7-300 有两个累加器而 S7-400 有 4 个。

（2）状态字寄存器

状态字寄存器是一个逐位定义的 16 位寄存器，用做程序运行状态的标志。它是一个程序可访问的寄存器，可按位和字节访问和设置，格式如图 4-6 所示。

15　9	8	7	6	5	4	3	2	1	0
—	BR	CC1	CC0	OS	OV	OR	STA	ROL	/FC

图 4-6　状态字寄存器格式

状态字的第 0 位为首先检查位（/FC），为 0 时表示逻辑网络开始或逻辑串的第一条指令，但在逻辑串指令执行过程中为 1，输出指令或结束逻辑串指令时清 0。

状态字的第 1 位为逻辑运算结果 ROL，为 0 时表示运算点无能流，为 1 时表示该点有能流，亦即存储的是逻辑运算前的结果。可用其来触发跳转指令。

状态字的第 2 位为状态位（STA），其值总是与执行位逻辑指令时该位的值保持一致。

状态字的第 3 位为或位（OR），除了在“与”“或”运算时，OR 位暂存“与”操作结果方便后面“或”外，OR 位均为 0。

状态字的第 4 位为溢出（OV）位，在指令执行出现溢出、非法操作、非法浮点数等错误时置 1，若后续指令运行正常则被清 0。

状态字的第 5 位为溢出状态保持（OS）位，当 OV 位为 1 时被置 1 后一直保持，用于记载之前运行指令时是否产生过错误。在执行三种指令（块调用或块结束以及 OS 为 1 时跳转的 JOS 指令）的情况下被复位为 0。

状态字的第 6 位和第 7 位为条件代码 0（CC0）和条件代码 1（CC1），两位条件代码用于表示在累加器 1 中运算结果与 0 的关系，比较指令的执行结果，移位指令的移出位状态。

执行指令后的 CC1 和 CC0 见表 4-6。

表 4-6 执行指令后的 CC1 和 CC0

CC1	CC0	算术运算无溢出	整数运算有溢出	浮点数运算有溢出	比较指令	移位指令和循环移位指令	字逻辑指令
0	0	结果 =0	相加下溢出	正负数绝对值过小	ACCU2 = ACCU1	移出位为 0	结果为 0
0	1	结果 <0	乘法下溢出，加减法上溢出	负数绝对值过大	ACCU2 < ACCU1	—	—
1	0	结果 >0	乘除法上溢出，加减法下溢出	正数上溢出	ACCU2 > ACCU1	—	结果不为 0
1	1	—	除法或 MOD 指令的除数为 0	非法的浮点数	非法的浮点数	移出位为 1	—

状态字的第 8 位为二进制结果位（BR），在具有位操作和字操作的一段程序中表示字操作结果正确与否。在梯形图中，BR 位与方框指令使能输出 ENO 一致，在编写 FB 和 FC 语句表程序中必须用 SAVE 指令将 RLO 存入 BR 来管理 BR 位，执行正确时为 1，错误时为 0。

状态字的第 9 ~ 15 位未赋值使用。

（3）地址寄存器（32 位）

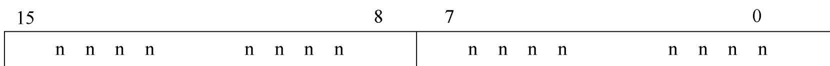
地址寄存器 AR1 和 AR2（32 位）的长度为 32 位，包括直接寻址指令的内部地址区或交叉地址区，存放完成寄存器间接寻址命令的区域内或区域间指针。其指针可访问存储区域 P、I、Q、M、DBX、DIX 和 L 的位、字节和双字。指针格式见寄存器间接寻址。

（4）存储器间接寻址

在存储器间接寻址中，指令要处理的数值单元（操作数所在地址）被放在地址指针所在的存储区域中，该存储区域必须是 M（位存储区）、DI（背景数据块）、DB（数据块）和 L（局部数据）中之一。

存储器间接寻址有字和双字两种指针格式，编号范围小于 65535 的定时器（T）、计数器（C）、数据块（DB）、功能块（FB）和功能（FC）用字指针，缩写以 W 结尾（如 DIW）。字指针格式如图 4-7 所示。

其他地址用双字指针格式，指针缩写以 D 结尾（如 DID）。双字指针格式如图 4-8 所示，位



位0~15 用于表示编号范围小于 65535 的定时器(T)、计数器(C)、数据块(DB)、功能块(FB)和功能(FC)

图 4-7　存储器间接寻址的字指针格式

0 ~ 2（xxx）为被寻址位的位编号（范围 0 ~ 7），位 3 ~ 18 为被寻址字节的字节编号（范围 0 ~ 65535）。用双字指针访问字、字节或双字存储器是指针位编号必须为 0。当存储器的间接地址存放在数据块中，调用时要先用 OPN 指令打开数据块。存储器间接寻址（黑体字部分）举例如下：

字指针格式　如　　A　T［DBW20］

```
OPN   DB50        //打开数据块 DB50
L   +30           //将整数 30 装入累加器 1
T   DBW20         //将累加器 1 内容传送至数据字 DBW20
A   T［DBW20］    //测试定时器 T30 的信号状态（字指针格式）
```

双字指针格式　L　QB［LD12 ］

```
L   P#5.0         //将 2#00000000000000000000000000101000 装入累加器 1
T   LD11          //将地址指针保存在区域数据双字 11 中
L   QB［LD11］    //将输出字节装入累加器 1，输出字节地址指针在区域双字 11 中，
                    装入的是 QB5（双字指针格式）
```

（5）寄存器间接寻址

S7 的寄存器间接寻址是指利用地址寄存器（AR1 或 AR2）的内容加上偏移量形成的地址指针，取得该指针所在单元存放的操作数的寻址方式。一个进行寄存器间接寻址的语句不改变地址寄存器中的数值。用寄存器指针访问一个字节、字或双字时，位编号必须为 0。

寄存器间接寻址的地址寄存器指针格式为双字指针，如图 4-8 所示，其中 0 ~ 2 位（xxx）为被寻址的位编号（0 ~ 7），第 3 ~ 18 位（bbbb bbbb bbbb bbbb）为被寻址的字节编号（0 ~ 65535），第 24 ~ 26 位（yyy）为被寻址地址的区域标识符，第 31 位 x 为地址区符号。

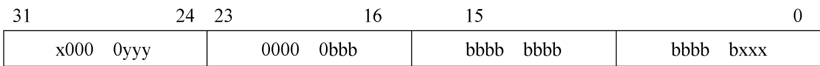


图 4-8　寄存器间接寻址的地址寄存器双字指针格式

地址寄存器包括直接寻址指令的内部地址区和交叉地址区，寄存器间接寻址因此分为区域内寄存器间接寻址和区域间寄存器间接寻址。其双字指针格式亦因此分为两种，由第 31 位来区分。

区域内寄存器间接寻址，第 31 位为 0，第 24 ~ 26 位（yyy）也为 0，存储区的类型在指令中给出，其指针可访问存储区域 P、I、Q、M、DBX、DIX 和 L 的位、字节和双字。双字指针格式与存储器间接寻址的双字指针格式一样。区域内寄存器间接寻址（黑体字部分）举例如下：

```
L   P#4.0                //将 2#0000000000000000 00000000 0010 0000 装入累加器 1
LAR1                     //将 ACCU1 中的内容送到地址寄存器 AR1 中
LAR2                     //将 ACCU1 中的内容送到地址寄存器 AR2 中
A   M［AR1，P#2.5］      //AR1 中的 P#4.0 加偏移量 P#2.5，亦即对 M6.5 进行操作，
```

```
                          AR1 中的内容不变
=  Q［AR1，P#0.6］        //逻辑运算的结果送 Q4.6，即 4.0（AR1）加 0.6（偏移量）
                          //AR1 中的的内容仍不变
L  DBW［AR1，P#18.0］     //将 DBW22 装入累加器 1，单元 22 来自 4（AR1）加 18（偏
                            移量），为 22
L  IB［AR1，P#50.0］      //将 IB54 装入累加器 1 ，单元 54 来自 4（AR1）加 50（偏移
                            量），为 54
L  LD［AR2，P#52.0］      //将 LD56 装入累加器 1，单元 56 来自 4（AR2）加 52（偏移
                            量），为 56
```

区域间寄存器间接寻址，第 31 位为 1，通过改变存储区域标识位 rrr 实现区域间跨区寻址。区域间寄存器间接寻址的区域标识位见表 4-7。

表 4-7　区域间寄存器间接寻址的区域标识位（yyy，第 24 ~ 26 位）

yyy(第 24 ~ 26 位)	区域标识(存储区)	yyy(第 24 ~ 26 位)	区域标识(存储区)
000	P(I/O,外设 I/O)	100	DBX(共享数据块)
001	I(输入过程映像)	101	DIX(背景数据块)
010	Q(输出过程映像)	110	L(局域数据)
011	M(位存储区)	111	VL(前导局域数据)

其指针可访问存储区域 P、I、Q、M、DBX、DIX 和 L 的位、字节和双字。存储区域由地址寄存器的第 24 ~ 26 位（yyy）确定。被寻址信息存放在地址寄存器中。区域间寄存器间接寻址举例如下：

```
L  P#I5.0             //将间接寻址的指针装入累加器 1
                      //指针为 2#1000 0001 0000 0000 0000 0000 0010 1000
LAR1                  //把指向位单元 M5.0 的双字指针从累加器 1 送至地址寄存器 1
L  P#Q5.0             //将间接寻址的位地址指针装入累加器 1
                      //地址指针为 2#1000 0001 0000 0000 0000 0000 0010 1000
LAR2                  //把指向位地址单元 I5.0 的双字指针从累加器 1 送地址寄存器 2
A  ［AR1，P#0.0］     //AR1 加偏移量 P#0.0，对输入位 I5.0 进行“与”操作
                      //AR1 的内容不变
=  ［AR2，P#1.1］     //AR2 加偏移量 P#1.1 逻辑运算结果（RLO）送 Q6.1
                      //AR2 的内容不变，Q6.1 来自 5.0（AR2）加上 1.1（偏移量）

L  P#I8.0             //将间接寻址的输入位地址指针装入累加器 1
                      //地址指针为 2#1000 0001 0000 0000 0000 0000 0100 0000
LAR2                  //把指向位地址单元 I8.0 的双字指针从累加器 1 送地址寄存器 2
L  P#M9.0             //将间接寻址的存储器的位 M9.0 的双字指针装入累加器 1
                      //地址指针为 2#1000 0011 0000 0000 0000 0000 0100 1000
LAR1                  //把指向存储器位 M9.0 的双字指针从累加器 1 送至地址寄存器 1
L  B［AR2，P#2.0］    //AR2 加偏移量 P#2.0，将输入字节 IB10 装入累加器 1AR1 的内
                        容不变
T  D［AR1，P#50.0］   //将累加器 1 的内容（输入字节 IB10）传送至存储双字 MD59，
```

累加器 1 中的输入字节来自 8（AR2）加 2（偏移量），为 10，
累加器 1 的内容为 1000 0001 0000 0000 0000 0000 0101 0000，
存储双字 MD58 来自 9（AR1）加 50（偏移量），为 59。

五、编程的一般规则

1. 网络

在梯形图（LAD）中，程序被分成称为网络的一些程序段，每个梯形图网络是由一个或多个梯级组成。功能块图（FBD）中，使用网络概念给程序分段。语句表（STL）程序中，使用“网络”这个关键词对程序分段。对梯形图、功能块图、语句表程序分段后，就可通过编程软件实现它们之间的相互转换。

2. 梯形图（LAD）/功能块图（FBD）

梯形图中左、右垂直线称为左、右母线。STEP7-Micro/Win32 梯形图编辑器在绘图时，通常将右母线省略。在左、右母线之间是由触点、线圈或功能框组合的有序排列。梯形图的输入总是在图形的左边，输出总是在图形的右边，因而触点与左母线相连，线圈或功能框终止右母线，从而构成一个梯级。在一个梯级中，左、右母线之间是一个完整的“电路”，不允许“短路”、“开路”，也不允许“能流”反向流动。

功能块图中输入总是在框图的左边，输出总是在框图的右边。

3. 允许输入端、允许输出端

在梯形图（LAD）、功能块图（FBD）中，功能框的 EN 端是允许输入端，功能框的允许输入端必须存在“能流”，即与之相连的逻辑运算结果为 1（即 EN = 1），才能执行该功能框的功能。

在语句表（STL）程序中没有 EN 允许输入端，但是允许执行 STL 指令的条件是栈顶的值必须是“1”。

在梯形图（LAD）、功能块图（FBD）中，功能框的 ENO 端是允许输出端，允许功能框的布尔量输出，用于指令的级联。

如果功能框允许输入端（EN）存在“能流”，且功能框准确无误地执行了其功能，那么允许输出端（ENO）将把“能流”传到下一个功能框，此时，ENO = 1。如果执行过程中存在错误，那么“能流”就在出现错误的功能框终止，即 ENO = 0。

在语句表（STL）程序中用 AENO（ANDENO）指令讯问，可以产生与功能框的允许输出端（ENO）相同的效果。

4. 条件/无条件输入

条件输入：在梯形图（LAD）、功能块图（FBD）中，与“能流”有关的功能框或线圈不直接与左母线连接。

无条件输入：在梯形图（LAD）、功能块图（FBD）中，与“能流”无关的功能框或线圈直接与左母线连接。例如 LBL、NEXT、SCR、SCRE 等。

5. 无允许输出端的指令

在梯形图（LAD）、功能块图（FBD）中，无允许输出端（ENO）的指令方框，不能用于级联。如 CALL SBR N（N1，…）子程序调用指令和 LBL、SCR 等。

第二节　S7-300/400 PLC 的指令系统

STEP7 有多种编程语言，本章着重介绍语句表（STL）和梯形图（LAD）这两种编程语言的

指令，并讨论基本指令的功能及编程方法。

一、位逻辑指令

位逻辑指令用于处理“0”和“1”两个二进制数逻辑运算，“1”表示编程元件动作或线圈通电，“0”表示编程元件未动作或线圈断电。

位逻辑指令根据布尔逻辑对扫描信号状态 0 和 1 进行处理，处理结果为 0 或 1，即位逻辑运算结果（RLO）。

1. 语句表（STL）的位逻辑指令

（1）布尔位逻辑指令

布尔位逻辑指令执行时检查被寻址位、定时器和计数器的信号状态以便确定激活“1”，去激活“0”，或确定置“1”或“0”。逻辑操作结果由状态字的第 0 位$\overline{FC}$确定，此位为 0 时，状态检查结果将保持不变，且不存于 RLO（表明逻辑串开始）；此位为 1 时，将检查结果与指令前的逻辑运算结果（RLO）和逻辑指令（A、O、X 等）进行逻辑运算后结果存入 RLO。

1）A“与”和 AN“与非”

指令格式为：A　< 位 >
　　　　　　AN　< 位 >

可寻址存储区为 I、Q、M、L、D、T 和 C。变量的数据类型为 BOOL、TIMER 和 COUNTER 型。A“与”表示串联的常开触点，AN“与非”表示串联的常闭触点。

例：

```
A   M0.0        //M0.0 为线圈的常开触点，信号为 1 时闭合通电，为 0 断电
AN  I0.0        //为常闭触点，信号为 1 时触点动作打开，为 0 时触点不动作
 =  Q4.0        //输出结束指令
```

2）O“或”和 ON“或非”

指令格式为：O　< 位 >
　　　　　　ON　< 位 >

可寻址存储区为 I、Q、M、L、D、T 和 C。变量的数据类型为 BOOL、TIMER 和 COUNTER 型。O“或”表示并联的常开触点，ON“或非”表示并联的常闭触点。

例：

```
O   M0.0        //M0.0 为线圈的常开触点，信号为 1 时闭合通电，为 0 断电
ON  I2.0        //为常闭触点，信号为 1 时触点动作打开，为 0 时触点不动作
```

3）X“异或”和 XN“异或非”

指令格式为：X　< 位 >
　　　　　　XN　< 位 >

可寻址存储区为 I、Q、M、L、D、T 和 C。变量的数据类型为 BOOL、TIMER 和 COUNTER 型。可连续使用“异或”指令。若检测到地址的信号状态为“1”不成对，逻辑运算的相互结果为“1”。

例 1：

```
X   I2.0
X   I2.1
 =  Q5.0        //输出 I2.0 和 I2.1 的“异或”结果
```

例 2：

```
X   I3.0
```

```
XN   I3.1
=    Q5.1              //输出 I3.0 和 I3.1 的“异或非”结果
```

4）O“先与后或”

指令格式为：O

O“先与后或”指令运算顺序为在“或”运算之前先进行“与”运算，对“与”运算执行逻辑“或”运算。

例：

```
A   I0.0
A   I0.1
O
A   I0.2
A   M0.2
=   Q4.0               //“先与后或”结果输出
```

（2）嵌套指令

指令格式为：A（　　“与”操作嵌套开始

AN（　　“与非”操作嵌套开始

O（　　“或”操作嵌套开始

ON（　　“或非”操作嵌套开始

X（　　“异或”操作嵌套开始

XN（　　“异或非”操作嵌套开始

）　　嵌套结束

程序执行时先执行嵌套括号内的逻辑运算，再做嵌套表达式前的逻辑操作。逻辑操作的结果是打开嵌套表达式的指令把上一个操作的 RLO 存放至嵌套堆栈，接着程序会将该存放的 RLO 与括号中逻辑操作中产生的结果进行组合，可以将 RLO 和 OR 位以及一个指令代码保存在嵌套堆栈中。指令程序允许最多 7 个嵌套堆栈输入项。使用）（嵌套闭合）指令，可删除嵌套堆栈中的一个输入项，并将状态字中的 OR 位复位，堆栈输入项中所包含的 RLO 与当前 RLO 相关，相关运算结果送入 RLO。

例：

```
A(
A     I0.0
A     I0.1
O
AN     I0.2
A     I0.3
)
AN     M8.0           //括号内运算完毕后和 M8.0 相“与”
=      Q4.0
```

（3）结束一个布尔位逻辑串指令

1）=赋值

指令格式为：=　<位>

寻址存储区为 I、Q、M、L 和 D。变量的数据类型为 BOOL 型。=（赋值指令）将其之前的

逻辑串语句的逻辑运算结果 RLO 写入指定的寻址位，作为寻址线圈的信号状态（亦即赋值指令 =赋值上一条语句的 RLO 来置位或复位寻址位）其特性是动态的。赋值指令结束一个逻辑串。

例 1：

```
A      I0.2
=      Q4.0
```

例 2：

```
A      I［MD5］
=      DBD［AR1，P#56.1］
```

2）S（置位）和 R（复位）指令

指令格式为：S <位>

R <位>

寻址存储区为 I、Q、M、L、D、T 和 C。变量的数据类型为 BOOL、TIMER 和 COUNTER 型。置位（S）和复位（R）指令用于把寻址位的信号状态置 1（变 1 并保持）和复位 0（变 0 并保持），具有静态特性。S 和 R 结束一个逻辑串。

当上一次逻辑运算结果 RLO =1，S 指令把寻址位的信号状态置 1，而 S 指令把寻址位的信号状态复位 0。此外，S 指令不能用于定时器但能给计数器置计数值，R 指令可用于定时器和计数器复位。

例 1：

```
A      I0.0
S      Q4.0
A      I0.1
R      Q4.0
```

例 2：

```
R      T22       //复位 T22
```

例 3：

```
L      C#77      //预置计数值装入累加器低字中
S      C22       //将预置值装入计数器 C22
R      C22       //复位 C22
```

（4）更改逻辑运算的结果（RLO）指令

1）NOT（RLO 取反）

指令格式为：NOT

使用 NOT 指令，对逻辑运算结果 RLO 取反。

例：

```
A（
A      I0.0
A      I0.1
O
AN     I0.2
A      I0.3
）
NOT
```

```
=      Q4.0        //括号内运算完毕 RLO 为 1 时，输出 Q4.0 为 0
```

2）SET（RLO 置位（=1））和 CLR（RLO 清零（=0））指令

指令格式为：SET

CLR

SET 和 CLR 指令对状态字的逻辑运算结果位 RLO 进行置位和复位操作，紧邻其后的赋值语句地址的信号状态随之变为 1 或 0 状态。

例：

```
SET                 //将 RLO 置位
 =      M0.0        //线圈 M0.0 通电
CLR                 //将 RLO 复位
 =      Q4.0        //线圈 Q4.0 断电
```

3）SAVE（把 RLO 存入 BR 寄存器）

指令格式为：SAVE

SAVE 指令的作用是把 RLO 存入二进制结果 BR 位，用来影响状态字的 BR 位或保存 RLO 以供将来需要。状态字的第 0 位首次检查位（/FC 位）在执行 SAVE 指令时不会被复位。编写 FB 和 FC 语句表程序中用 SAVE 管理 BR 位，执行正确时使 RLO 为 1，存入 BR 位，错误时 BR 存入 0。通常在检查相同块或附属块中的 BR 位之前不用 SAVE 指令，其原因是状态字的第 0 位首次检查位（/FC 位）在执行 SAVE 指令时不会被复位，BR 位的状态将参与下一个网络的“与”逻辑运算，在保存和检查操作之间，BR 位可能被许多指令修改过。

一般在退出逻辑块之前使用 SAVE 指令，使能输出 ENO（即 BR 位）被设置为 RLO 位的值，用于对块中的错误进行检查。

（5）RLO 脉冲边沿触发指令 FN（下降沿）和 FP（上升沿）

指令格式为：FN　<位>

FP　<位>

寻址存储区为 I、Q、M、L 和 D。变量的数据类型为 BOOL 型。

下降沿检测指令（FN <位>）检测到 RLO 的下降沿（RLO 的信号状态 从“1”变为“0”时）后，RLO 位为“1”，能流在一个扫描周期内流过检测元件。

上升沿检测指令（FP <位>）检测到 RLO 的上升沿（RLO 的信号状态 从“0”变为“1”时）后，RLO 位为“1”，能流在一个扫描周期内流过检测元件。

在每一个程序扫描周期过程中，RLO 位的信号状态都将与前一周期中获得的结果进行比较，看信号状态是否有变化。前一 RLO 的信号状态必须保存在边沿标志地址（<位>）中，以进行比较。如果在当前和先前的 RLO 状态之间有变化（检测到下降沿或上升沿），则在操作之后，能流在该扫描周期内流过检测元件，亦即 RLO 位仅在该扫描周期内为“1”；如果在当前和先前的 RLO 状态之间没有变化（无脉冲边沿），则在操作之后，FN 和 FP 指令均把 RLO 复位为 0。

例 1：

```
A      I0.0
A      I0.1
FN   M0.0        //若检测到下降沿，
 =      Q4.0     //则 Q4.0 仅在一个 OB1 扫描周期得电
```

例 2：

```
A      I0.0
A      I0.1
```

FP　M0.0　　　//若检测到上升沿，
=　　Q4.0　　　//则 Q4.0 仅在一个 OB1 扫描周期得电

2. 梯形图（LAD）的位逻辑指令

（1）—| |—常开触点（地址）、—| / |—常闭触点（地址）和—（ ）输出线圈

指令格式为：　< 地址 >　　　　　　< 地址 >　　　　　　< 地址 >

—| |—　　　　　　—| / |—　　　　　　—（ ）

常开触点和常闭触点的可寻址存储区为 I、Q、M、L、D、T 和 C，其变量的数据类型为 BOOL、TIMER 和 COUNTER 型。

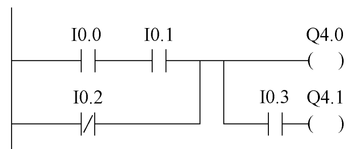
输出线圈的可寻址存储区为 I、Q、M、L、D，其变量的数据类型为 BOOL 型。

常开触点对应的地址位为 1 状态时，该触点闭合，信号能流流经触点，逻辑运算结果（RLO）= "1"；常开触点对应的地址位为 0 状态时，该触点打开，无信号能流流经触点，逻辑运算结果（RLO）= "0"。串联使用时常开触点通过"与（AND）"逻辑链接到 RLO 位。并联使用时，常开触点通过"或（OR）"逻辑链接到 RLO 位。

常闭触点对应的地址位为 0 状态时，该触点闭合，信号能流流经触点，逻辑运算结果（RLO）= "1"；常闭触点对应的地址位为 1 状态时，该触点打开，无信号能流流经触点，逻辑运算结果（RLO）= "0"。串联使用时常闭触点通过"与（AND）"逻辑链接到 RLO 位。并联使用时，常闭触点通过"或（OR）"逻辑链接到 RLO 位。

输出线圈的工作方式与继电器逻辑图中线圈的工作方式类似。当有能流通过线圈（RLO = 1），将 < 地址 > 位置"1"。若没有能流通过线圈（RLO = 0），将 < 地址 > 位置"0"。只能将输出线圈置于梯级的右端。可以有多个（最多 16 个）输出单元。使用 -| NOT |—（能流取反）单元可以创建取反输出。—| NOT |—（能流取反）指令取 RLO 位的非值。

例：



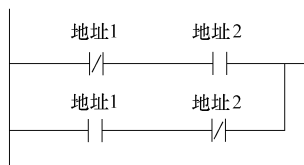
满足条件输入端 I0.0 和 I0.1 的信号状态为"1"时或输入端 I0.2 的信号状态为"0"时。输出端 Q4.0 的信号状态将是"1"。

满足条件输入端 I0.0 和 I0.1 的信号状态为"1"时，或输入端 I0.2 的信号状态为"0"、输入端 I0.3 的信号状态为"1"时，输出端 Q4.1 的信号状态将是"1"。

如果实例梯级在激活的 MCR 区之内：MCR 处于接通状态时，将按照上述能流状态置位 Q4.0 和 Q4.1。MCR 处于断开状态（=0）时，无论是否有能流通过，都将 Q4.0 和 Q4.1 复位为 0。

（2）XOR　逻辑"异或"

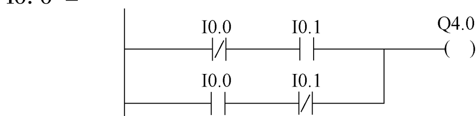
指令格式：



寻址存储区为 I、Q、M、L、D、T 和 C。当两个指定位的信号状态不同时，RLO 为"1"。

例：

如果（I0.0 = “0” 且 I0.1 = “1”）或者（I0.0 = “1” 且 I0.1 = “0”），输出 Q4.0 将是“1”。



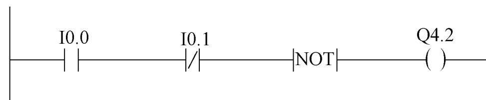
（3）—| NOT |—能流取反

指令格式为：—| NOT |—

能流取反用来将它左边电路的逻辑运算结果 RLO 取反，该运算结果若为 1 则变为 0，为 0 则变为 1，该指令没有操作数。亦即能流到达该触点时即停止流动；若能流未到达该触点，该触点给右侧供给能流。

例：

满足 I0.0 的信号状态为“1”且 I0.1 的信号状态为“0”时，输出端 Q4.2 的信号状态将是“0”。



（4）—(#)—中间输出

指令格式为：　<地址>
　　　　　　　—(#)—

寻址存储区为 I、Q、M、D、L（只有在逻辑块（FC、FB、OB）的变量声明表中将 L 区地址声明为 TEMP 时，才能使用 L 区地址）。

—(#)—（中间输出）是中间分配单元，它将 RLO 位状态（能流状态）保存到指定<地址>。中间输出单元保存前面分支单元的逻辑结果。以串联方式与其他触点连接时，可以像插入触点那样插入—(#)—。不能接在左侧的垂直“电源线”上，也不能放在电路最右端结束的位置。使用—| NOT |—（能流取反）单元可以创建取反—(#)—。中间输出指令等效图如图 4-9 所示。

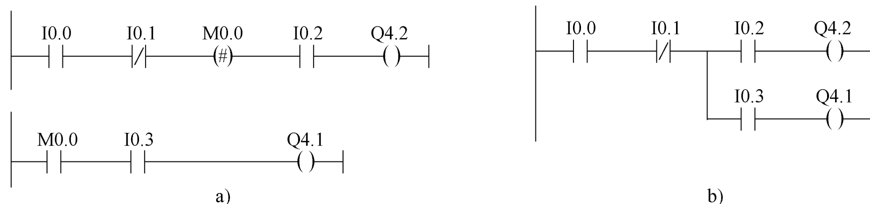


图 4-9　中间输出指令等效图

（5）—(R) 复位线圈和—(S) 置位线圈

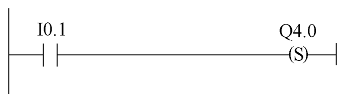
指令格式为：　<地址>　　　　　　　<地址>
　　　　　　　—(R)　　　　　　　　(S)

复位指令寻址存储区为 I、Q、M、L、D、T、C，而置位指令寻址存储区为 I、Q、M、L、D（能流通过线圈）时，—(R) 指令将指定的地址位复位（变为 0 并保持），<地址>也可以是值复位为“0”的定时器（T 编号）或值复位为“0”的计数器（C 编号）。—(S) 指令则将指定的地址位置位（变为 1 并保持）。

例：



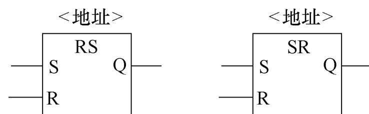
I0.0 的信号状态为“1”时 T1 的复位为“0”。



I0.1 的信号状态为“1”时 Q4.0 置位为“1”。

（6）RS 触发器和 SR 触发器

指令格式为：

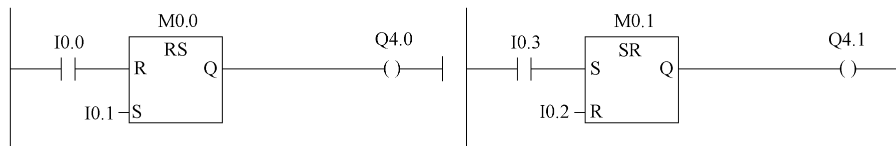


寻址存储区为 I、Q、M、L、D，数据类型为 BOOL 型。

若 R 端（S 端）为“1”，S 端（R 端）为“0”，则复位 RS 触发器（置位 SR 触发器）。否则，如果 R 端（S 端）为“0”，S 端（R 端）为“1”，则置位 RS 触发器（复位 SR 触发器）。

如果两个输入端（R 端和 S 端）的 RLO 均为“1”，RS 触发器先在指定 <地址> 执行复位指令，然后执行置位指令，以使该地址在执行余下的程序扫描过程中保持置位状态。SR 触发器先在指定 <地址> 执行置位指令，然后执行复位指令，以使该地址在执行余下的程序扫描过程中保持复位状态。

例：



当 I0.0 为“1”，I0.1 为“0”，则复位存储器位 M0.0，输出 Q4.0 位为“0”。当 I0.3 为“1”，I0.2 为“0”，则置位存储器位 M0.0，输出 Q4.1 位为“1”。

（7）--(N)-- RLO 下降沿检测和--(P)-- RLO 上升沿检测

指令格式为： <地址> <地址>

--(N)-- ---(P)---

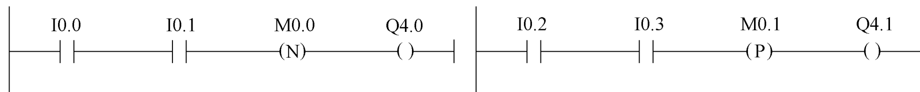
寻址存储区为 I、Q、M、L、D，数据类型为 BOOL 型。

下降沿检测指令（--(N)--）检测到 RLO 的下降沿（RLO 的信号状态 从“1”变为“0”时）后，RLO 位为“1”，能流在一个扫描周期内流过检测元件。

上升沿检测指令（--(P)--）检测到 RLO 的上升沿（RLO 的信号状态 从“0”变为“1”时）后，RLO 位为“1”，能流在一个扫描周期内流过检测元件。

在每一个程序扫描周期过程中，RLO 位的信号状态都将与前一周期中获得的结果进行比较，看信号状态是否有变化。前一 RLO 的信号状态必须保存在边沿标志地址（<地址>）中，以进行比较。如果在当前和先前的 RLO 状态之间有变化（检测到下降沿或上升沿），则在操作之后，能流在该扫描周期内流过检测元件，亦即 RLO 位仅在该扫描周期内为“1”；如果在当前和先前的 RLO 状态之间没有变化（无脉冲边沿），则在操作之后，RLO 边沿检测指令均把 RLO 复位为 0。

例：



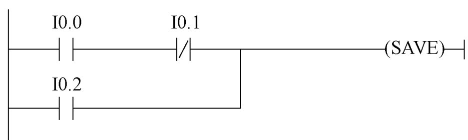
M0.0 位检测到下降沿时，Q4.0 通电。 M0.1 位检测到下降沿时，Q4.1 通电。

（8）--(SAVE) 将 RLO 状态保存到 BR 寄存器

指令格式为： ---(SAVE)

此指令将 RLO 保存到状态字的 BR 位，但不复位第一个校验位 /FC。因此，BR 位的状态将包含在下一程序段的 AND 逻辑运算中。一般在使用 SAVE 后不在同一块或从属块中校验 BR 位，因为这期间执行的指令中有许多会对 BR 位进行修改。建议在退出块前使用 SAVE 指令，因为 ENO 输出（ = BR 位）届时已设置为 RLO 位的值，所以可以检查块中是否有错误。

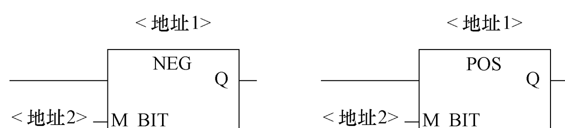
例：



将梯级（ = RLO）的状态保存到 BR 位。

（9）NEG　地址下降沿检测和 POS　地址上升沿检测

指令格式为：



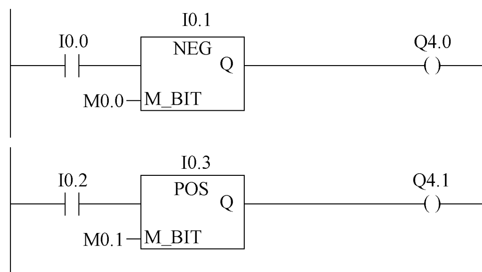
寻址存储区为 I、Q、M、L、D，数据类型为 BOOL 型。< 地址 1 > 为已扫描信号， < 地址 2 > 为 M_ BIT 边沿存储位，存储 < 地址 1 > 的前一个信号状态，Q 为单触发输出。

NEG（地址下降沿检测）比较 < 地址 1 > 的信号状态与前一次扫描的信号状态（存储在 < 地址 2 > 中）。如果当前 RLO 状态为“0”且其前一状态为“1”（检测到下降沿），执行此指令后 RLO 位将是“1”。而 POS（地址上升沿检测）比较 < 地址 1 > 的信号状态与前一次扫描的信号状态（存储在 < 地址 2 > 中）。如果当前 RLO 状态为“1”且其前一状态为“0”（检测到上升沿），执行此指令后 RLO 位将是“1”。

例：

I0. 0 的信号状态为“1”且 I0. 1 有下降沿时 Q4. 0 的信号状态为“1”。

I0. 2 的信号状态为“1”且 I0. 3 有上升沿时 Q4. 1 的信号状态为“1”。



二、比较指令

1. 语句表（STL）的比较指令

比较指令用于比较累加器 ACCU1 和 ACCU2 中相同数据类型的两个数据的大小，比较条件成立时 RLO 为 1，否则为 0。用于比较的数据类型有三种：整数、长整数和浮点数。比较的关系有六种，即 = = （ACCU1 等于 ACCU2）、< >（ACCU1 不等于 ACCU2）、 > （ ACCU1 大于 ACCU2）、<（ACCU1 小于 ACCU2）、 > =（ACCU1 大于等于 ACCU2）、 < =（ACCU1 小于等于 ACCU2）。状态字位 CC 1 和 CC 0 用于表示关系“小于”、“等于”或“大于”。

？I 比较整数（16 位）、？D　比较长整数（32 位）和 ？R　比较浮点数（32 位）

指令格式为：

= = I、< > I、> I、< I、> = I、< = I　= = D、

< > D、> D、< D、> = D、< = D

= = R、< > R、> R、< R、> = R、< = R

比较整数（16 位）指令将 ACCU 2-L 的内容与 ACCU 1-L 的内容进行比较。ACCU 2-L 和 ACCU 1-L 的内容被解释为 16 位整数。比较长整数（32 位）指令将 ACCU 2 的内容与 ACCU 1 的内容进行比较。ACCU 2 和 ACCU 1 的内容被解释为 32 位整数。比较浮点数（32 位，IEEE-FP）指令将 ACCU 2 的内容与 ACCU 1 的内容进行比较。ACCU 1 和 ACCU 2 的内容被解释为浮点数（32 位，IEEE-FP）。

其比较结果由 RLO 和相关状态字位的设置来表示。RLO = 1 表示比较结果为 true；RLO = 0 表示比较结果为 false。状态字位 CC 1 和 CC 0 表示关系“小于”、“等于”或“大于”。

例 1　比较整数：

```
L   MW10    //装载 MW10 的内容（16 位整数）
L   IW24    //装载 IW24 的内容（16 位整数）
>I          //比较 ACCU 2-L（MW10）是否大于（>）ACCU 1-L（IW24）
=M 2.0      //RLO = 1（如果 MW10 > IW24）
```

例 2　比较长整数：

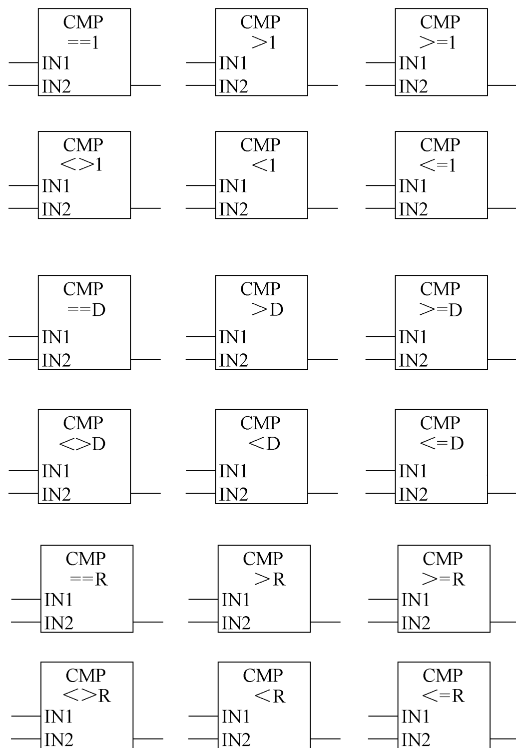
```
L   MD10    //装载 MD10 的内容（长整型，32 位）
L   ID24    //装载 ID24 的内容（长整型，32 位）
>D          //比较 ACCU 2（MD10）是否大于（>）ACCU 1（ID24）
=M 2.0      //RLO = 1（如果 MD10 > ID24）
```

例 3　比较浮点数：

```
L   MD10          //装载 MD10 的内容（浮点数）
L   1.359E+02     //装载常数 1.359E+02
>R                //比较 ACCU 2（MD10）是否大于（>）ACCU 1（1.359-E+02）
=M 2.0            //RLO = 1（如果 MD10 > 1.359E+02）
```

2. 梯形图（LAD）的比较指令

比较指令用于比较 IN1 和 IN2 中相同数据类型的两个数据的大小，比较条件成立时 RLO 为 1，否则为 0。用于比较的数据类型有三种：整数、长整数和浮点数（实数）。比较的关系有六种，即 = =（IN1 等于 IN2）、< >（IN1 不等于 IN2）、>（IN1 大于 IN2）、<（IN1 小于 IN2）、> =（IN1 大于或等于 IN2）、< =（IN1 小于或等于 IN2）。如果以串联方式使用比较单元，则使用“与”运算将其链接至梯级程序段的 RLO；如果以并联方式使用该框，则使用“或”运算将其链接至梯级程序段的 RLO。比较指令的使用方法与标准触点类似，它可位于任何可放置标准触点的位置。

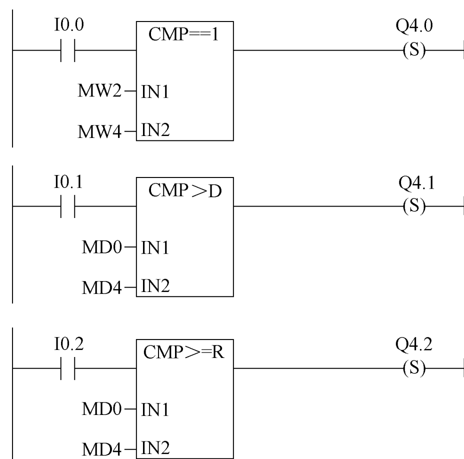


CMP ? I 整数比较、CMP ? D 长整数比较和 CMP ? R 实数比较

指令符号为：

寻址存储区为 I、Q、M、L、D。

例：



整数比较：输入 I0.0 的信号状态为“1”且 MW2 > = MW4 时输出 Q4.0 置位。

长整数比较：输入 I0.1 的信号状态为“1”且 MD0 > MD4 时输出 Q4.1 置位。

实数比较：输入 I0.1 的信号状态为“1”且 MD0 > = MD4 时输出 Q4.2 置位。

三、转换指令

数据转换指令将累加器 1 中的数据进行数据类型的转换，转换的结果仍然在累加器 1 中。寻址存储区均为 I、Q、M、L、D。

1. 语句表（STL）的转换指令

（1）BTI　将 BCD 码转换为整型（16 位）

指令格式为：BTI

BTI（3 位 BCD 数从十进制到二进制的转换）将 ACCU 1-L（累加器的低字，下同）中的 3 位二进制编码的十进制数（BCD 码）转换为 16 位整型，结果存储在累加器 1 的低字中，累加器 1 的高字和累加器 2 则保持不变。

ACCU 1-L 中的 BCD 数字：BCD 数字的允许值范围从“－999”～“＋999”。位 0～位 11 为数值，位 15 为 BCD 数字的符号（0 = 正，1 = 负）。位 12～位 14 在转换中不使用。如果 BCD 数字的十进制（四位）数字处于 10～15 的无效范围，则在转换期间会出现错误代码 B#16#21（表示出现 BCD 码转换错误）。通常，CPU 会转入 STOP 模式。但是，通过对 OB121 编程可设计另一种出错响应，用以处理该同步编程错误。

例：

```
L   MW10      //将 BCD 数字载入 ACCU 1-L
BTI           //从 BCD 码转换为整型，将结果存储在 ACCU 1-L 中
T   MW20      //将结果（整数）传送到 MW20
```

（2）ITB　将整型（16 位）转换为 BCD 码

指令格式为：ITB

ITB（16 位整数从二进制到十进制的转换）将 ACCU 1-L 的 16 位整数转换为 3 位二进制编码的十进制数（BCD 码），结果存储在累加器 1 的低字中。位 0～位 11 包含 BCD 数字的值。位 12～位 15 用来表示 BCD 数字的符号状态（0000 = 正，1111 = 负）。累加器 1 的高字和累加器 2 保持不变。BCD 数字的范围为“－999”～“＋999”。如果超出允许范围，则状态位 OV 和 OS 被置位为 1。执行该指令时不涉及 RLO，也不会影响 RLO。

例：

```
L   MW10  //将整数载入 ACCU 1-L
ITB       //从整型转换为 BCD 码（16 位），结果存储在 ACCU 1-L 中
T   MW20  //将结果（BCD 数字）传送到 MW20
```

（3）BTD 将 BCD 码转换为整型（32 位）

指令格式为：BTD

BTD（7 位 BCD 数字从十进制到二进制的转换）将 ACCU 1 中的 7 位二进制编码的十进制数（BCD），转换为 32 位长整型，结果存储在累加器 1 中，累加器 2 则保持不变。ACCU 1 中的 BCD 数字：BCD 数字的允许值范围从“－9，999，999”～“＋9，999，999”。位 0～位 27 解释为数

值，位 31 解释为 BCD 数字的符号（0 = 正，1 = 负）。位 28 ~ 位 30 在转换中不使用。如果任何十进制数（BCD 编码的四位组）处于 10 ~ 15 的无效范围，则在转换期间会出现 BCDF 错误。通常，CPU 会转入 STOP 模式。但是，通过对 OB121 编程可设计另一种出错响应，用以处理该同步编程错误。

例：

```
L   MD10    //将 BCD 数字载入 ACCU 1
BTD         //从 BCD 码转换为整型，将结果存储在 ACCU 1 中
T   MD20    //将结果（长整数）传送到 MD20
```

（4）ITD 将整型（16 位）转换为长整型（32 位）

指令格式为：ITD

ITD（16 位整数转换为 32 位整数）将 ACCU 1-L 中的 16 位整数转换为 32 位长整数，结果存储在累加器 1 中。

例 ：

```
L   MW12    //将整数载入 ACCU 1
ITD         //从整型（16 位）转换为长整型（32 位），结果存储在 ACCU 1 中
T   MD20    //将结果（长整型）传送到 MD20
```

（5）DTB 将长整型（32 位）转换为 BCD 码

指令格式为：DTB

DTB（32 位整数从二进制到十进制的转换）将 ACCU 1 中的 32 位长整型转换为 7 位二进制编码的十进制数（BCD 码），结果存储在累加器 1 中。位 0 ~ 位 27 包含 BCD 数字的值。位 28 ~ 位 31 用来表示 BCD 数字的符号状态（0000 ＝ 正，1111 ＝ 负）。累加器 2 保持不变。

BCD 数的范围为“－9，999，999” ~ “＋9，999，999”。如果超出允许范围，则状态位 OV 和 OS 被置位为 1。

例 ：

```
L   MD10    //将 32 位整数载入 ACCU 1
DTB         //从整型（32 位）转换为 BCD 码，结果存储在 ACCU 1 中
T   MD20    //将结果（BCD 数字）传送到 MD20
```

（6）DTR　将长整型（32 位）转换为浮点型（32 位 IEEE-FP）

指令格式为：DTR

DTR（32 位整数转换为 32 位 IEEE 浮点数）将 ACCU 1 中的 32 位长整型转换为 32 位 IEEE 浮点数。如必要，该指令会对结果取整。32 位整数比 32 位浮点数精度更高，结果存储在累加器 1 中。

例：

```
L   MD10    //将 32 位整数载入 ACCU 1
DTR         //从长整型转换为浮点型（32 位 IEEE FP），结果存储在 ACCU 1 中
T   MD20    //将结果（BCD 数字）传送到 MD20
```

（7）INVI　　对整数（16 位）求反码

指令格式为：INVI

INVI（对整数求反码）在 ACCU 1-L 中形成 16 位数值的二进制反码。二进制反码是通过将各个位的值取反形成的，即用“0”替换“1”，用“1”替换“0”。结果存储在累加器 1 的低字中。

例：

```
L   IW8     //将值载入 ACCU 1-L
```

INVI　　//形成 16 位二进制反码

T　MW10//将结果传送到 MW10

（8）INVD　　对长整数（32 位）求反码

指令格式为：INVD

INVD（对长整数求反码）在 ACCU1 中形成 32 位数值的二进制反码。二进制反码是通过将各个位的值取反形成的，即用“0”替换“1”，用“1”替换“0”。结果存储在累加器 1 中。

例：

L　ID8　//将值载入 ACCU 1

INVD　　//形成二进制反码（32 位）

T　MD10//将结果传送到 MD10

（9）NEGI　　对整数（16 位）求补码

指令格式为：NEGI

NEGI（对整数求补码）在 ACCU 1-L 中形成 16 位数值的二进制补码。二进制补码是通过将各个位取反（即用“0”替换“1”，用“1”替换“0”）后加“1”形成的。结果存储在累加器 1 的低字中。二进制补码相当于原码乘以“－1”。状态位 CC 1、CC 0、OS 和 OV 则根据函数运算的结果来设置。

例：

L　IW8　　//将值载入 ACCU 1-L

NEGI　　//形成 16 位二进制补码

T　MW10//将结果传送到 MW10

（10）NEGD　　对长整数（32 位）求补码

指令格式为：NEGD

NEGD（对长整数求补码）在 ACCU 1 中形成 32 位数值的二进制补码。二进制补码是通过将每个位取反（即用“0”替换“1”，用“1”替换“0”）后加“1”形成的，结果存储在累加器 1 中。二进制补码指令相当于原码乘以“－1”。执行该指令时不涉及 RLO，也不影响 RLO。状态位 CC 1、CC 0、OS 和 OV 则根据函数运算的结果来设置。

例：

L　ID8　//将值载入 ACCU 1 中

NEGD　　//生成二进制补码（32 位）

T　MD10//将结果传送到 MD10

（11）NEGR　　对浮点数（32 位，IEEE-FP）取反

指令格式为：NEGR

NEGR（将 32 位 IEEE 浮点数取反）将 ACCU 1 中的浮点数（32 位，IEEE-FP）取反。该指令将 ACCU 1 中位 31 的状态（尾数符号）取反，结果存储在累加器 1 中。

例：

L　ID8　//将值载入 ACCU 1（实例：ID 8 ＝ 1. 5E＋02）

NEGR　　//对浮点数（32 位，IEEE-FP）取反，结果存储在 ACCU 1 中

T　MD10//将结果传送到 MD10（实例：结果 ＝ －1. 5E＋02）

（12）CAW 改变 ACCU 1-L（16 位）中的字节顺序

指令格式为：CAW

CAW 反转 ACCU 1-L 中的字节顺序，即交换 ACCU 1-L 中的 2B 的位置。结果存储在累加器 1

的低字中，累加器 1 的高字和累加器 2 则保持不变。

例：

```
L   MW10   //将 MW10 的值载入 ACCU 1
CAW        //反转 ACCU 1-L 中的字节顺序
T   MW20   //将结果传送到 MW20
```

执行前后变化如下：

内容	ACCU1-H-H	ACCU1-H-L	ACCU1-L-H	ACCU1-L-L
执行 CAW 之前	值 A	值 B	值 C	值 D
执行 CAW 之后	值 A	值 B	值 D	值 C

（13）CAD 改变 ACCU 1（32 位）中的字节顺序

指令格式为：CAD

CAD 反转 ACCU 1 中的字节顺序。结果存储在累加器 1 中，累加器 2 则保持不变。

例：

```
L   MD10 //将 MD10 的值载入 ACCU 1
CAD      //反转 ACCU 1 中的字节顺序
T   MD20 //将结果传送到 MD20
```

执行前后变化如下：

内容	ACCU1-H-H	ACCU1-H-L	ACCU1-L-H	ACCU1-L-L
执行 CAD 之前	值 A	值 B	值 C	值 D
执行 CAD 之后	值 D	值 C	值 B	值 A

（14）RND 取整

指令格式为：RND

RND（32 位 IEEE 浮点数转换为 32 位整型）将 ACCU 1 中的 32 位 IEEE 浮点数（32 位，IEEE-FP）转换为 32 位整型（长整型），并将结果取整为最接近的整数。如果所转换数字的小数部分介于偶数和奇数结果之间，则该指令选择偶数结果。如果数字超出允许范围，则状态位 OV 和 OS 被置位到 1。结果存储在累加器 1 中。出现错误（使用了不能表示为 32 位整数的 NaN 或浮点数）时不执行转换并显示溢出。

例：

```
L    MD10   //将浮点数载入 ACCU 1-L
RND         //将浮点数（32 位，IEEE-FP）转换为整型（32 位）并对结果进行舍入
T    MD20   //将结果（长整数）传送到 MD20
```

执行前后变化如下：

转换前的值　　　　　　转换后的值

MD10 = “100.5” => RND => MD20 = “+100”

MD10 = “-100.5” => RND => MD20 = “-100”

（15）TRUNC 截尾取整

指令格式为：TRUNC

TRUNC（32 位 IEEE 浮点数转换为 32 位整型）将 ACCU 1 中的 32 位 IEEE 浮点数转换为 32 位整型（长整型）。运算结果为所转换浮点数的整数部分（IEEE 取整模式“取整到零”），小数部分舍去。如果数字超出允许范围，则状态位 OV 和 OS 被置位到 1。结果存储在累加器 1 中。出现错误（使用了不能表示为 32 位整数的 NaN 或浮点数）时不执行转换并显示溢出。

例：

L　MD10　//将浮点数载入 ACCU 1-L

TRUNC　　//将浮点数（32 位）转换为整型（32 位），并对结果取整后储存在 ACCU 1 中

T　MD20　//将结果（长整数）传送到 MD20

执行前后变化如下：

转换前的值　　　　　转换后的值

MD10　=　“100.5”　=＞　TRUNC　=＞　MD20　=　“＋100”

MD10　=　“－100.5”　=＞　TRUNC　=＞　MD20　=　“－100”

（16）RND＋取整为高位长整数

指令格式为：RND＋

RND＋（32 位 IEEE 浮点数转换为 32 位整型）将 ACCU 1 中的 32 位 IEEE 浮点数转换为 32 位整型（长整型），并将结果取整为大于或等于所转换浮点数的最小整数，亦即将浮点数转换为大于等于它的最小长整数。如果数字超出允许范围，则状态位 OV 和 OS 被置位到 1。结果存储在累加器 1 中。出现错误（使用了不能表示为 32 位整数的 NaN 或浮点数）时不执行转换并显示溢出。

例：

L　MD10　//将浮点数（32 位，IEEE-FP）载入 ACCU 1-L 中

RND＋　　//将浮点数转换为整型（32 位），并对结果取整，将输出存储在 ACCU 1 中

T　MD20　//将结果（长整数）传送到 MD20

执行前后变化如下：

转换前的值　　　　　　转换后的值

MD10　=　“100.5”　=＞　RND＋　=＞MD20　=　“＋101”

MD10　=　“－100.5”　=＞　RND＋　=＞MD20　=　“－100”

（17）RND-取整为低位长整数

指令格式为：RND-

RND-（32 位 IEEE 浮点数转换为 32 位整型）将 ACCU 1 中的 32 位 IEEE 浮点数转换为 32 位整型（长整型），并将结果取整为小于或等于所转换浮点数的最大整数，亦即将浮点数转换为小于等于它的最小长整数。如果数字超出允许范围，则状态位 OV 和 OS 被置位到 1。结果存储在累加器 1 中。出现错误（使用了不能表示为 32 位整数的 NaN 或浮点数）时不执行转换并显示溢出。

例：

L　MD10　//将浮点数载入 ACCU 1-L

RND-　　//将浮点数转换为整型（32 位），并对结果取整。结果存储在 ACCU 1 中

T　MD20　//将结果（长整数）传送到 MD20

执行前后变化如下：

转换前的值　　　　　转换后的值

MD10　=　″100.5″　=＞　RND－　=＞　MD20　=　″＋100″

MD10　=　″－100.5″　=＞　RND－　=＞　MD20　=　″－101″

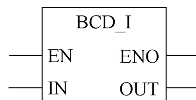
2. 梯形图（LAD）的比较指令

转换指令读取参数 IN 的内容，然后进行转换或改变其符号。可通过参数 OUT 查询结果。寻址存储区均为 I、Q、M、L、D。参数 EN 为起用输入，ENO 为起用输出，IN 为待转换的值，

OUT 为转换结果。

（1）BCD_ I　BCD 码转换为整型

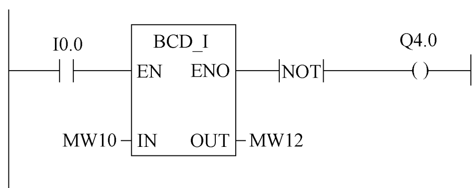
指令符号为：



BCD_ I（BCD 码转换为整型）将参数 IN 的内容以 3 位 BCD 码数字（+/-999）读取，并将其转换为整型值（16 位）。整型值的结果通过参数 OUT 输出。ENO 始终与 EN 的信号状态相同。

例：

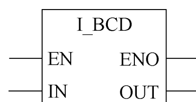
如果输入 I0.0 的状态为“1”，则将 MW10 中的内容以 3 位 BCD 码数字读取，并将其转换为整型值。结果存储在 MW12 中。如果未执行转换（ENO = EN = 0），则输出 Q4.0 的状态为“1”。



（2）I_ BCD　整型转换为 BCD 码

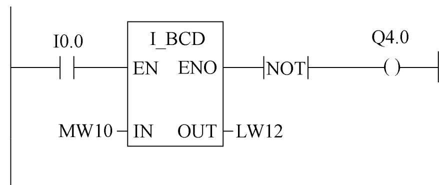
指令符号为：

I_ BCD（整型转换为 BCD 码）将参数 IN 的内容以整型值（16 位）读取，并将其转换为 3 位 BCD 码数字（+/- 999）。结果由参数 OUT 输出。如果产生溢出，ENO 的状态为“0”。



例：

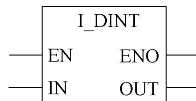
如果 I0.0 的状态为“1”，则将 MW10 的内容以整型值读取，并将其转换为 3 位 BCD 码数字。结果存储在 LW12 中。如果产生溢出或未执行指令（I0.0 = 0），则输出 Q4.0 的状态为“1”。



（3）I_ DINT　整型转换为长整型

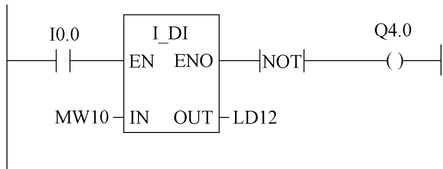
指令符号为：

I_ DINT（整型转换为长整型）将参数 IN 的内容以整型（16 位）读取，并将其转换为长整型（32 位）。结果由参数 OUT 输出。ENO 始终与 EN 的信号状态相同。



例：

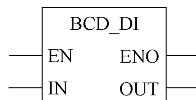
如果 I0.0 为“1”，则 MW10 的内容以整型读取，并将其转换为长整型。结果存储在 LD12 中。如果未执行转换（ENO = EN = 0），则输出 Q4.0 的状态为“1”。



（4）BCD_ DI　BCD 码转换为长整型

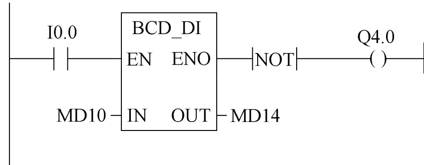
指令符号为：

BCD_ DI（将 BCD 码转换为长整型）将参数 IN 的内容以 7 位 BCD 码（+/- 9999999）数字读取，并将其转换为长整型值（32 位）。长整型值的结果通过参数 OUT 输出。ENO 始终与 EN 的信号状态相同。



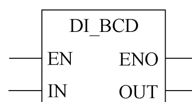
例：

如果 I0.0 的状态为“1”，则将 MD10 的内容以 7 位 BCD 码数字读取，并将其转换为长整型值。结果存储在 MD14 中。如果未执行转换（ENO = EN = 0），则输出 Q4.0 的状态为“1”。



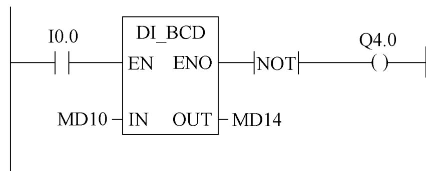
（5）DI_ BCD　长整型转换为 BCD 码

指令符号为：



DI_ BCD（长整型转换为 BCD 码）将参数 IN 的内容以长整型值（32 位）读取，并将其转换为七位 BCD 码数字（+/-9999999）。结果由参数 OUT 输出。如果产生溢出，ENO 的状态为“0”。

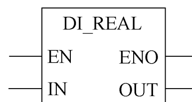
例：



如果 I0.0 的状态为“1”，则将 MD10 的内容以长整型读取，并将其转换为 7 位 BCD 码数字。结果存储在 MD14 中。如果产生溢出或未执行指令（I0.0 = 0），则输出 Q4.0 的状态为“1”。

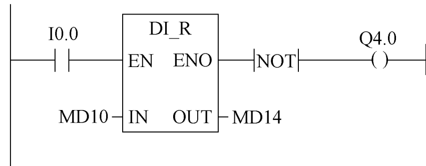
（6）DI_ REAL　长整型转换为浮点型

指令符号为：



DI_ REAL（长整型转换为浮点型）将参数 IN 的内容以长整型读取，并将其转换为浮点数。结果由参数 OUT 输出。ENO 始终与 EN 的信号状态相同。

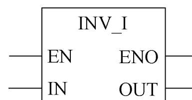
例：



如果 I0.0 的状态为“1”，则将 MD10 中的内容以长整型读取，并将其转换为浮点数。结果存储在 MD14 中。如果未执行转换（ENO = EN = 0），则输出 Q4.0 的状态为“1”。

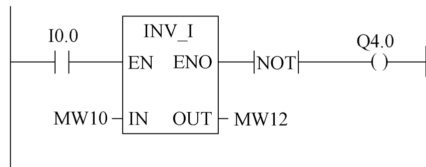
（7）INV_ I　对整数求反码

指令符号为：



INV_ I（对整数求反码）读取 IN 参数的内容，并使用十六进制掩码 W#16#FFFF 执行布尔“异或”运算。此指令将每一位变成相反状态。ENO 始终与 EN 的信号状态相同。

例：

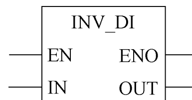


如果 I0.0 为“1”，则将 MW8 的每一位都取反，例如：MW10 = 01000001 10000001 取反结果为 MW12 = 10111110 01111110。

如果未执行转换（ENO = EN = 0），则输出 Q4.0 的状态为“1”。

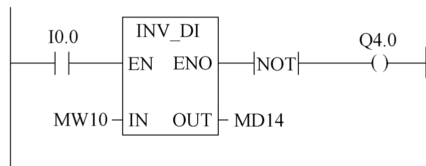
（8）INV_ DI　对长整数求反码

指令符号为：



INV_ DI（对长整数求反码）读取 IN 参数的内容，并使用十六进制掩码 W#16#FFFF FFFF 执行布尔“异或”运算。此指令将每一位转换为相反状态。ENO 始终与 EN 的信号状态相同。

例：



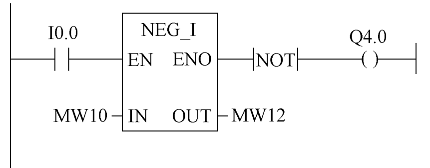
如果 I0.0 为“1”，则 MD8 的每一位都取反，例如：MD10 = F0FF FFF0 取反结果为 MD14 = 0F00 000F。如果未执行转换（ENO = EN = 0），则输出 Q4.0 的状态为“1”。

（9）NEG_ I　对整数求补码

指令符号为：

NEG_ I（对整数求补码）读取 IN 参数的内容并执行求二进制补码指令。二进制补码指令等同于乘以（－1）后改变符号（例如：从正值变为负值）。ENO 始终与 EN 的信号状态相同，以下情况例外：如果 EN 的信号状态 ＝1 并产生溢出，则 ENO 的信号状态 ＝0。

例：



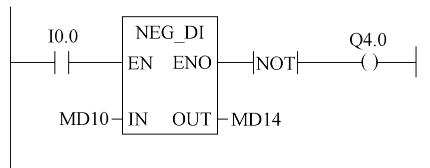
如果 I0.0 为“1”，则由 OUT 参数将 MW10 的值（符号相反）输出到 MW12。MW8＝＋10 结果为 MW10＝－10。如果未执行转换（ENO＝EN＝0），则输出 Q4.0 的状态为“1”。如果 EN 的信号状态 ＝1 并产生溢出，则 ENO 的信号状态 ＝0。

（10）NEG_ DI　对长整数求补码

指令符号为：

NEG_ DI（对长整数求补码）读取参数 IN 的内容并执行二进制补码指令。二进制补码指令等同于乘以（－1）后改变符号（例如：从正值变为负值）。ENO 始终与 EN 的信号状态相同，以下情况例外：如果 EN 的信号状态 ＝1 并产生溢出，则 ENO 的信号状态 ＝0。

例：



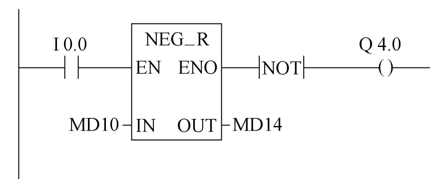
如果 I0.0 为“1”，则由 OUT 参数将 MD10 的值（符号相反）输出到 MD14。MD8 ＝ ＋1000 结果为 MD12＝－1000。如果未执行转换（ENO＝EN＝0），则输出 Q4.0 的状态为“1”。如果 EN 的信号状态＝1 并产生溢出，则 ENO 的信号状态＝0。

（11）NEG_ R　浮点数取反

指令符号为：

NEG_ R（取反浮点）读取参数 IN 的内容并改变符号。指令等同于乘以（－1）后改变符号（例如：从正值变为负值）。ENO 始终与 EN 的信号状态相同。

例：



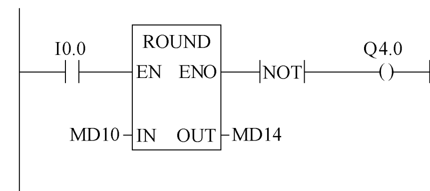
如果 I0.0 为“1”，则由 OUT 参数将 MD10 的值（符号相反）输出到 MD14。MD10 ＝ ＋5.234 结果为 MD14 ＝ －5.234。如果未执行转换（ENO ＝ EN ＝0），则输出 Q4.0 的状态为“1”。

（12）ROUND　取整为长整型

指令符号为：

ROUND（取整为长整型）将参数 IN 的内容以浮点数读取，并将其转换为长整型（32 位）。结果为最接近的整数（“取整到最接近值”）。如果浮点数介于两个整数之间，则返回偶数。结果由参数 OUT 输出。如果产生溢出，ENO 的状态为“0”。

例：



如果 I0.0 的状态为“1”，则将 MD10 中的内容以浮点数读取，并将其转换为最接近的长整数。函数“取整为最接近值”的结果存储在 MD14 中。如果产生溢出或

未执行指令（I0.0 = 0），则输出 Q4.0 的状态为“1”。

（13）TRUNC　截取长整数部分

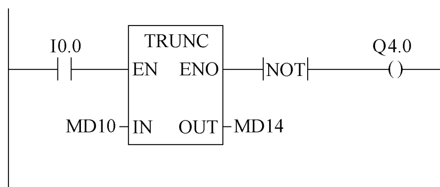
指令符号为：

TRUNC
EN　ENO
IN　OUT

TRUNC（截断长整型）将参数 IN 的内容以浮点数读取，并将其转换为长整型（32 位）。(“向零取整模式”）的长整型结果由参数 OUT 输出。如果产生溢出，ENO 的状态为“0”。

例：

如果 I0.0 的状态为“1”，则将 MD10 中的内容以实型数字读取，并将其转换为长整型值。结果为浮点数的整型部分，并存储在 MD14 中。如果产生溢出或未执行指令（I0.0 = 0），则输出 Q4.0 的状态为“1”。



（14）CEIL 向上取整

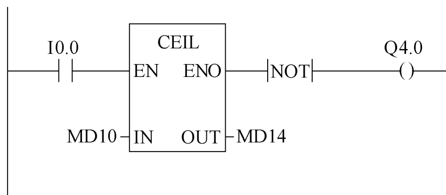
指令符号为：

CEIL
EN　ENO
IN　OUT

CEIL（向上取整）将参数 IN 的内容以浮点数读取，并将其转换为长整型（32 位）。结果为大于该浮点数的最小整数（“取整到 + 无穷大”）。如果产生溢出，ENO 的状态为“0”。

例：

如果 I0.0 为 1，则将 MD10 的内容以浮点数读取，并使用取整函数将其转换为长整型。结果存储在 MD14 中。如果出现溢出或未处理指令（I0.0 = 0），则输出 Q4.0 的状态为“1”。



（15）FLOOR　向下取整

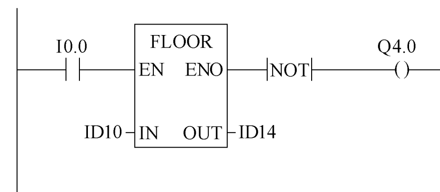
指令符号为：

FLOOR
EN　ENO
IN　OUT

FLOOR（向下取整）将参数 IN 的内容以浮点数读取，并将其转换为长整型（32 位）。结果为小于该浮点数的最大整数部分（“取整为 - 无穷大”）。如果产生溢出，ENO 的状态为“0”。

例：

如果 I0.0 为“1”，则将 ID10 的内容以浮点数读取，并按取整到负无穷大模式将其转换为长整型。结果存储在 ID14 中。如果产生溢出或未执行指令（I0.0 = 0），则输出 Q4.0 的状态为“1”。



四、计数器指令

计数器是 STEP 7 编程语言中用于计数的功能单元。在 CPU 存储器中，有为计数器保留的区域。此存储区为每个计数器保留一个 16 位字。语句表指令集和梯形图指令集均支持 256 个计数器。计数器指令是仅有的可访问计数器存储区的函数。计数值的范围为 0 ~ 999，计数字的 0 ~ 11 位是计数值的 BCD 码。

1. 语句表（STL）的计数器指令

（1）FR 起用计数器（释放）

指令格式为：　FR <计数器>

可寻址存储区为 C。变量的数据类型为 COUNTER 型。当 RLO 从“0”跳转到“1”时，FR <计数器>会将边缘检测标记清零，该标记用于设置和选择寻址计数器的升值或降值计数。设置

计数器或进行正常计数时不需起用计数器。也就是说，即使设置计数器的预设值、升值计数器或降值计数器的常数 RLO 为 1，在起用之后也不会再次执行这些指令。

例：

A　I2.0　//检查输入 I 2.0 的信号状态

FR　C3　//当 RLO 由 0 变为 1 时，起用计数器 C3

（2）L　　将当前计数器值载入 ACCU 1

指令格式为：L <计数器>

可寻址存储区为 C。变量的数据类型为 COUNTER 型。ACCU 1 的内容保存到 ACCU 2 中后，L <计数器> 将寻址计数器的当前计数值作为整型值载入 ACCU 1-L。

例：

L　　C3　//以二进制格式将计数器 C3 的计数值载入 ACCU 1-L

（3）LC　　将当前计数器值作为 BCD 码载入 ACCU 1

指令格式为：LC <计数器>

可寻址存储区为 C。变量的数据类型为 COUNTER 型。ACCU 1 的原有内容保存到 ACCU 2 中后，LC <计数器>将寻址计数器的计数值作为 BCD 码载入 ACCU 1。

例：

LC　C3　//以二进制编码的十进制格式（BCD）将计数器 C3 的计数值载入 ACCU 1-L

（4）R　　将计数器复位

指令格式为：R <计数器>

可寻址存储区为 C。变量的数据类型为 COUNTER 型。如果 RLO = 1，R <计数器> 会将“0”载入寻址计数器中。

例：

A　I2.3　//检查输入 I 2.3 的信号状态

R　C3　//如果 RLO 从 0 变到 1，则将计数器 C3 复位到 0

（5）S　　设置计数器预设值

指令格式为：S <计数器>

可寻址存储区为 C。变量的数据类型为 COUNTER 型。当 RLO 从“0”跳转到“1”时，S <计数器> 将 ACCU 1-L 的计数值载入寻址计数器。ACCU 1 中的计数值必须是介于“0”和“999”之间的 BCD 码形式的数值。

例：

A　I2.3　//检查输入 I 2.3 的信号状态

L　C#3　//将计数值 3 载入 ACCU 1-L

S　C1　//如果 RLO 从 0 跳转到 1，则设置计数器 C1，以进行计数

（6）CU　　升值计数器

指令格式为：CU <计数器>

寻址存储区为 C。变量的数据类型为 COUNTER 型。当 RLO 从“0”跳转到“1”，并且计数小于“999”时，CU <计数器>将寻址计数器的计数值增 1。当计数值达到上限“999”时，升值过程停止。RLO 的附加跳转无效，而且溢出的 OV 位不被置位。

例：

A　I2.1　//如果在输入 I 2.1 有上升沿改变

CU　C3　//当 RLO 从 0 跳转到 1 时，计数器 C3 的计数值增 1

（7）CD　　降值计数器

指令格式为：CD　<计数器>

寻址存储区为 C。变量的数据类型为 COUNTER 型。当 RLO 从“0”跳转到“1”，并且计数大于 0 时，CD　<计数器>将寻址计数器的计数值减 1。当计数达到下限“0”时，降值过程停止。由于计数器不用负值计数，所以 RLO 的附加跳转无效。

例：

```
L     C#14     //计数器预设值
A     I0.1     //检测到 I 0.1 的上升沿后，预置计数器的值
S     C1       //如果起用预置，则载入计数器 1 预设值
A     I0.0     //每当 I 0.0 有上升沿时，降值计数一次
CD    C1       //当 RLO 根据输入 I 0.0 的状态从 0 跳转至 1 时，将计数器 C1 减 1
AN    C1       //使用 C1 位进行零检测
=Q 0.0         //如果计数器 1 的值为零，则 Q 0.0 = 1
```

2. 梯形图（LAD）的计数器指令

（1）S _ CUD　双向计数器

指令符号为：

C no.
S_CUD
CU　Q
CD
S　CV
PV　CV_BCD
R

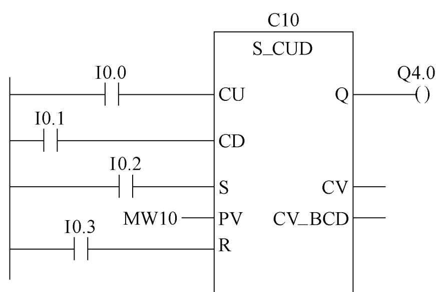
Cno. 为计数器标识号，寻址存储区为 C。

CU 为升值计数输入；CD 为降值计数输入；S 为预设计数器设置输入；PV 为预设计数器的值，将计数器值以“C#<值>”的格式输入（范围为 0 ~ 999）；R 为复位输入；CV 为当前计数器值，十六进制数字；CV_ BCD 为当前计数器值；BCD 码 Q 为计数器状态，寻址存储区均为 I、Q、M、L、D。

如果输入 S 有上升沿，S_ CUD（双向计数器）预置为输入 PV 的值。如果输入 R 为 1，则计数器复位，并将计数值设置为零。如果输入 CU 的信号状态从“0”切换为“1”，并且计数器的值小于“999”，则计数器的值增 1。如果输入 CD 有上升沿，并且计数器的值大于“0”，则计数器的值减 1。如果两个计数输入都有上升沿，则执行两个指令，并且计数值保持不变。如果已设置计数器，并且输入 CU/CD 的 RLO = 1，则即使没有从上升沿到下降沿或下降沿到上升沿的切换，计数器也会在下一个扫描周期进行相应的计数。如果计数值大于等于零（“0”），则输出 Q 的信号状态为“1”。避免在多个程序点使用同一计数器（可能出现计数出错）。

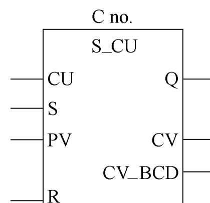
例：

如果 I0.2 从“0”变为“1”，则计数器预设为 MW10 的值。如果 I0.0 的信号状态从“0”改变为“1”，则计数器 C10 的值将增加 1，当 C10 的值等于“999”时除外。如果 I0.1 从“0”改变为“1”，则 C10 减少 1，但当 C10 的值为“0”时除外。如果 C10 不等于零，则 Q4.0 为“1”。



（2）S_ CU　升值计数器

指令符号为：



Cno. 为计数器标识号，寻址存储区为 C。

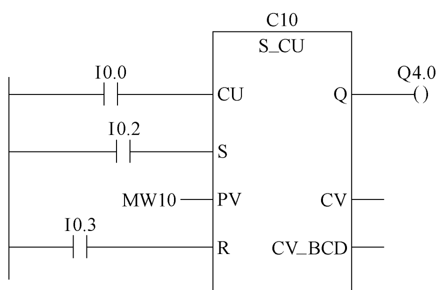
CU 为升值计数输入；S 为为预设计数器设置输入；PV 为预设计数器的值，将计数器值以“C#<值>”的格式输入（范围为 0 ~ 999）；R 为复位输入；CV 为当前计数器值，十六进制数字；CV_ BCD 为当前计数器值；

BCD 码 Q 为计数器状态，寻址存储区均为 I、Q、M、L、D。

如果输入 S 有上升沿，则 S_ CU（升值计数器）预置为输入 PV 的值。如果输入 R 为“1”，则计数器复位，并将计数值设置为零。如果输入 CU 的信号状态从“0”切换为“1”，并且计数器的值小于“999”，则计数器的值增 1。如果已设置计数器，并且输入 CU 的 RLO = 1，则即使没有从上升沿到下降沿或下降沿到上升沿的切换，计数器也会在下一个扫描周期进行相应的计数。如果计数值大于等于零（“0”），则输出 Q 的信号状态为“1”。

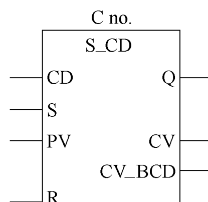
例：

如果 I0.2 从“0”改变为“1”，则计数器预置为 MW10 的值。如果 I0.0 的信号状态从“0”改变为“1”，则计数器 C10 的值将增加 1，当 C10 的值等于“999”时除外。如果 C10 不等于零，则 Q4.0 为“1”。



（3）S_ CD 降值计数器

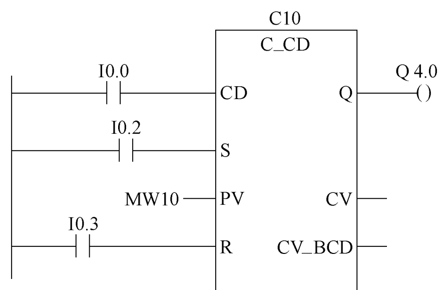
指令符号为：



如果输入 S 有上升沿，则 S_ CD（降值计数器）设置为输入 PV 的值。如果输入 R 为 1，则计数器复位，并将计数值设置为零。如果输入 CD 的信号状态从“0”切换为“1”，并且计数器的值大于零，则计数器的值减 1。如果已设置计数器，并且输入 CD 的 RLO = 1，则即使没有从上升沿到下降沿或下降沿到上升沿的改变，计数器也会在下一个扫描周期进行相应的计数。如果计数值大于等于零（“0”），则输出 Q 的信号状态为“1”。

例：

如果 I0.2 从“0”改变为“1”，则计数器预置为 MW10 的值。如果 I0.0 的信号状态从“0”改变为“1”，则计数器 C10 的值将减 1，当 C10 的值等于“0”时除外。如果 C10 不等于零，则 Q4.0 为“1”。



（4）---（ SC ） 设置计数器值

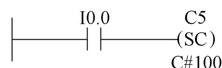
指令符号为： <C 编号>

---（ SC ）

<C 编号>为要预置的计数器编号。<预设值>的寻址存储区为 I、Q、M、L、D 或常数，预置 BCD 的值（0 ~ 999）。---（ SC ）（设置计数器值）仅在 RLO 中有上升沿时才会执行。此时，预设值被传送至指定的计数器。

例：

如在 I0.0 有上升沿（从“0”改变为“1”），则计数器 C5 预置为 100。如果没有上升沿，则计数器 C5 的值保持不变。



（5）---（ CU ） 升值计数器线圈

指令符号为：<C 编号>

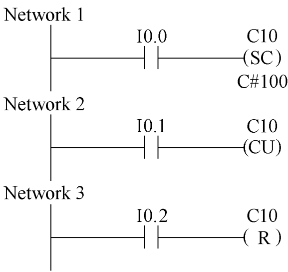
---（ CU ）

<C 编号>为要预置的计数器编号，其范围依赖于 CPU，如在 RLO 中有上升沿，并且计数器的值小于“999”，则---（ CU ）（升值计数器线圈）将指定计数器的值加 1。如果 RLO 中没有

上升沿，或者计数器的值已经是“999”，则计数器值不变。

例：

如果输入 I0.0 的信号状态从“0”改变为“1”（RLO 中有上升沿），则将预设值 100 载入计数器 C10。如果输入 I0.1 的信号状态从“0”改变为“1”（在 RLO 中有上升沿），则计数器 C10 的计数值将增加 1，但当 C10 的值等于“999”时除外。如果 RLO 中没有上升沿，则 C10 的值保持不变。如果 I0.2 的信号状态为“1”，则计数器 C10 复位为“0”。



（6）---（ CD ）　降值计数器线圈

指令符号为：<C 编号>

---（ CD）

如果 RLO 状态中有上升沿，并且计数器的值大于“0”，则---（ CD ）（降值计数器线圈）将指定计数器的值减 1。如果 RLO 中没有上升沿，或者计数器的值已经是“0”，则计数器值不变。

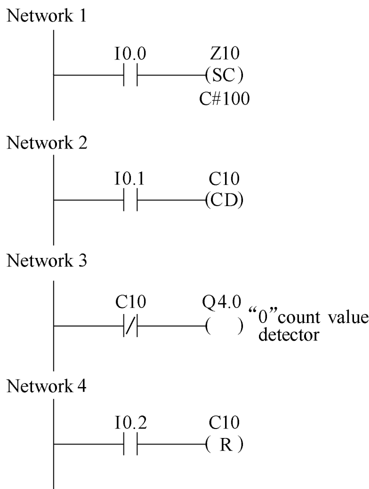
例：

如果输入 I0.0 的信号状态从“0”改变为“1”（RLO 中有上升沿），则将预设值 100 载入计数器 C10。

如果输入 I0.1 的信号状态从“0”改变为“1”（在 RLO 中有上升沿），则计数器 C10 的计数值将减 1，但当 C10 的值等于“0”时除外。如果 RLO 中没有上升沿，则 C10 的值保持不变。

如果计数值 = 0，则接通 Q4.0。

如果输入 I0.2 的信号状态为“1”，则计数器 C10 复位为“0”。



五、数据块指令

“打开数据块”（OPN）指令将数据块作为共享数据块或背景数据块打开。同时只能打开一个共享数据块或一个背景数据块，访问已经打开的数据块内的存储单元时，其地址中不必指明是哪一个数据块的数据单元。例如在打开 DB10 后，DB10. DBW32 可简写为 DBW32 。

1. 语句表（STL）的数据块指令

（1）OPN　　打开数据块

指令格式为：OPN <数据块>

数据块类型为 DB、DI，源地址为 1 ~ 65535。OPN <数据块> 将数据块作为共享数据块或背景数据块打开。可以同时打开一个共享数据块和一个背景数据块。

例：

```
OPN   DB10     //将数据块 DB10 作为共享数据块打开
L     DBW35    //将已打开数据块的数据字 35 装载到 ACCU 1-L 中
T     MW22     //将 ACCU 1-L 的内容传送到 MW22 中
OPN   DI20     //将数据块 DB20 作为背景数据块打开
L     DIB12    //将已打开背景数据块的数据字节 12 装载到 ACCU 1-L 中
```

T　　DBB37　　//将 ACCU 1-L 的内容传送到已打开的共享数据块的数据字节 37

（2）CDB　　交换共享数据块和背景数据块

指令格式为：CDB

CDB 用于交换共享数据块和背景数据块。该指令的作用是交换数据块寄存器。共享数据块变为背景数据块，或相反。

（3）L DBLG　　在 ACCU 1 中装载共享数据块的长度

指令格式为：L DBLG

L DBLG（装载共享数据块的长度）会在 ACCU 1 的内容保存到 ACCU 2 中后，将共享数据块的长度装载到 ACCU 1 中。

例：

OPN　DB10　//将数据块 DB10 作为共享数据块打开

L　　DBLG　//装载共享数据块的长度（DB10 的长度）

L　　MD10　//如果数据块足够长，则该长度作为用于比较的值

< D

JC　　ERRO　//如果长度小于 MD10 中的值，则跳转至 ERRO 跳转标签

（4）L DBNO　　在 ACCU 1 中装载共享数据块的编号

指令格式为：L DBNO

L DBNO（装载共享数据块的编号）会在 ACCU 1 的内容保存到 ACCU 2 中后，将打开的共享数据块的编号装载到 ACCU 1-L 中。

（5）L DILG　　在 ACCU 1 中装载背景数据块的长度

指令格式为：L DILG

L DILG（装载背景数据块的长度）会在 ACCU 1 的内容保存到 ACCU 2 中后，将背景数据块的长度装载到 ACCU 1-L 中。

例：

OPN　D120　//将 DB20 数据块作为背景数据块打开

L　　DILG　//装载背景数据块的长度（DB20 的长度）

L　　MW10　//如果数据块足够长，则该长度作为用于比较的值

< 1

JC　　　　//如果长度小于 MW10 中的值，则跳转到 ERRO 跳转标签

（6）L　DINO　　在 ACCU 1 中装载背景数据块的编号

指令格式为：L DINO

L DINO（装载背景数据块的编号）会在 ACCU 1 的内容保存到 ACCU 2 中后，将打开的背景数据块的编号装载到 ACCU 1 中。

2. 梯形图（LAD）的数据块指令---（OPN）打开数据块：DB 或 DI

指令符号为：< DB 编号 > 或 < DI 编号 >

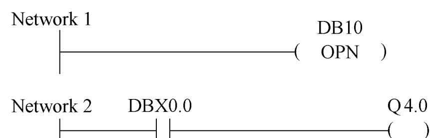
---（OPN）

---（OPN）（打开数据块）打开共享数据块（DB）或情景数据块（DI）。---（OPN）函数是一种对数据块的无条件调用。将数据块的编号传送到 DB 或 DI 寄存器中。后续的 DB 和 DI 命令根据寄存器内容访问相应的块。

例：

打开数据块 10（DB10）。触点地址（DBX0.0）引用包含在 DB10 中的当前数据记录的数据

字节零的第零位。将此位的信号状态分配给输出 Q4.0。



六、逻辑控制指令

所有逻辑块（组织块（OB）、功能块（FB）和功能（FC））中均可使用逻辑控制指令。可使用跳转指令来控制逻辑流，允许程序中断其线性流，在一个不同点处继续进行扫描。可使用 LOOP 指令来多次调用一个程序段。

跳转指令的地址是标签。标签最多可以包含 4 个字符。第一个字符必须是字母表中的字母，其他字符可以是字母或数字（例如，SEG3）。跳转标签指示程序将要跳转到的目标。在语句表中跳转标签后必须带有一个冒号“:”，且在行中必须位于程序语句之前。在梯形图中目标标签必须位于程序段的开头。可以通过从梯形图浏览器中选择 LABEL，在程序段的开头输入目标标签。在显示的空框中，键入标号的名称。

1. 语句表（STL）的逻辑控制指令

（1）JU　无条件跳转

指令格式为：JU ＜跳转标签＞

JU ＜跳转标签＞中断线性程序扫描，并跳转到一个跳转目标，与状态字的内容无关。线性程序扫描在跳转目标处继续执行。由跳转标签确定跳转目标。允许向前跳转和向后跳转。只能在一个块内执行跳转，即跳转指令和跳转目标必须位于同一个块内。跳转目标在该块内必须唯一。最大跳转距离为程序代码的 −32768 或 +32767 个字。可以跳过的实际语句的最大数目取决于程序中使用的语句组合（单字、双字或三字语句）。

例：

```
A     I1.0
A     I1.2
JC    DELE    //当 RLO=1 时，跳转到跳转标签 DELE
L     MB10
INC   1
T     MB10
JU    FORW    //无条件跳转到跳转标签 FORW
DELE：L0
T      MB10
FORW：A   I2.1//跳转到跳转标签 FORW 后，在此继续执行程序扫描
```

（2）JL　跳转到标签（多分支跳转）

指令格式为：JL ＜跳转标签＞

JL ＜跳转标签＞（通过跳转到列表进行跳转）允许编程多次跳转。跳转目标列表（最多 255 个条目）从 JL 指令的下一行开始，到 JL 地址中引用的跳转标签的前一行结束。每个跳转目标由一个 JU 指令组成。跳转目标的数目（0～255）则从 ACCU 1-L-L 中获取。

只要 ACCU 的内容小于 JL 指令和跳转标签之间跳转目标的数目，JL 指令就跳转到 JU 指令中的一条。当 ACCU 1-L-L=0 时，跳转到第一条 JU 指令。当 ACCU 1-L-L=1 时，跳转到第二个 JU 指令，以此类推。如果跳转目标的数目太大，则在跳转到目标列表中的最后一条 JU 指令后，JL 指令跳转到第一条指令处。

跳转目标列表必须包含 JU 指令，该指令位于 JL 指令地址中引用的跳转标签之前。跳转列表内的所有其他指令都是非法的。

例：

```
L  MB0      //将跳转目标数目装载到 ACCU 1-L-L 中
JL  LSTX    //当 ACCU 1-L-L > 3 时的跳转目标
JU  SEG0    //当 ACCU 1-L-L = 0 时的跳转目标
JU  SEG1    //当 ACCU 1-L-L = 1 时的跳转目标
JU  COMM    //当 ACCU 1-L-L = 2 时的跳转目标
JU  SEG3    //当 ACCU 1-L-L = 3 时的跳转目标
LSTX：JU COMM
SEG0：*//允许的指令
 *
JU        COMM
SEG1：*//允许的指令
 *
JU        COMM
SEG3：*//允许的指令
 *
JU        COMM
COMM：*
 *
```

（3）JC　　当 RLO = 1 时跳转

指令格式为：JC <跳转标签>

当逻辑运算的结果为 1 时，JC <跳转标签>就中断线性程序扫描，并跳转到一个跳转目标。线性程序扫描在跳转目标处继续执行。由跳转标签确定跳转目标。允许向前跳转和向后跳转。只能在一个块内执行跳转，即跳转指令和跳转目标必须位于同一个块内。跳转目标在该块内必须唯一。最大跳转距离为程序代码的 -32768 或 +32767 个字。可以跳过的实际语句的最大数目取决于程序中使用的语句组合（单字、双字或三字语句）。当逻辑运算的结果为 0 时，不执行跳转。将 RLO 设置为 1，继续对下一个语句执行程序扫描。

例：

```
A   I 1.0
A   I 1.2
JC  JOVR       //当 RLO=1 时，跳转到跳转标签 JOVR
L   IW8        //当不执行跳转时，在此继续执行程序扫描
T   MW22
JOVR：A I 2.1  //跳转到跳转标签 JVOR 后，在此继续执行程序扫描
```

（4）JCN　　当 RLO = 0 时跳转

指令格式为：JCN <跳转标签>

当逻辑运算的结果为 0 时，JCN <跳转标签>就中断线性程序扫描，并跳转到一个跳转目标。线性程序扫描在跳转目标处继续执行。由跳转标签确定跳转目标。允许向前跳转和向后跳转。只能在一个块内执行跳转，即跳转指令和跳转目标必须位于同一个块内。跳转目标在该块内必须唯一。最大跳转距离为程序代码的 -32768 或 +32767 个字。可以跳过的实际语句的最大数目取决于程序中使用的语句组合（单字、双字或三字语句）。当逻辑运算的结果为 1 时，不执行

跳转。继续对下一个语句执行程序扫描。

例：

```
A      I 1.0
A      I 1.2
JCN    JOVR      //当 RLO = 0 时跳转到跳转标签 JOVR
L      IW8       //当不执行跳转时，在此继续执行程序扫描
T      MW22
JOVR：A I 2.1  //跳转到跳转标签 JOVR 后，在此继续执行程序扫描
```

（5）JCB　当带 BR 位的 RLO = 1 时跳转

指令格式为：JCB <跳转标签>

当逻辑运算的结果为 1 时，JCB <跳转标签>就中断线性程序扫描，并跳转到一个跳转目标。线性程序扫描在跳转目标处继续执行。由跳转标签确定跳转目标。允许向前跳转和向后跳转。只能在一个块内执行跳转，即跳转指令和跳转目标必须位于同一个块内。跳转目标在该块内必须唯一。最大跳转距离为程序代码的 -32768 或 +32767 个字。可以跳过的实际语句的最大数目取决于程序中使用的语句组合（单字、双字或三字语句）。当逻辑运算的结果为 0 时，不执行跳转。将 RLO 设置为 1，继续对下一个语句执行程序扫描。将 RLO 复制到 BR 中，以执行 JCB <跳转标签>指令，而与 RLO 的状态无关。

例：

```
A      I1.0
A      I1.2
JCB    JOVR      // RLO = 1 时，跳转到标签 JOVR。将 RLO 位的内容复制到 BR 位
L      IW8       //当不执行跳转时，在此继续执行程序扫描
T      MW22
JOVR：A   I 2.1//跳转到跳转标签 JOVR 后，在此继续执行程序扫描
```

（6）JNB　　当带 BR 位的 RLO = 0 时跳转

指令格式为：JNB <跳转标签>

当逻辑运算的结果为 0 时，JNB <跳转标签>就中断线性程序扫描，并跳转到一个跳转目标。线性程序扫描在跳转目标处继续执行。由跳转标签确定跳转目标。允许向前跳转和向后跳转。只能在一个块内执行跳转，即跳转指令和跳转目标必须位于同一个块内。跳转目标在该块内必须唯一。最大跳转距离为程序代码的 -32768 或 +32767 个字。可以跳过的实际语句的最大数目取决于程序中使用的语句组合（单字、双字或三字语句）。当逻辑运算的结果为 1 时，不执行跳转。将 RLO 设置为 1，继续对下一个语句执行程序扫描。当存在一个 JNB <跳转标签>指令时，将 RLO 复制到 BR，而与 RLO 的状态无关。

例：

```
A      I 1.0
A      I 1.2
JNB    JOVR      //当 RLO = 0 时，跳转到标签 JOVR。将 RLO 位的内容复制到 BR 位中
L      IW8       //当不执行跳转时，在此继续执行程序扫描
T      MW22
JOVR：AI 2.1   //跳转到跳转标签 JOVR 后，在此继续执行程序扫描
```

（7）JBI　　当 BR = 1 时跳转

指令格式为：JBI <跳转标签>

当状态位 BR 为 1 时，JBI <跳转标签>就中断线性程序扫描，并跳转到一个跳转目标。线性程序扫描在跳转目标处继续执行。跳转目标由跳转标签确定。跳转标签最多 4 个字符，且第一个字符必须为字母。跳转标签后必须带有一个冒号“：”，且在行中必须位于程序语句之前。允许向前跳转和向后跳转。只能在一个块内执行跳转，即，跳转指令和跳转目标必须位于同一个块内。跳转目标在该块内必须唯一。最大跳转距离为程序代码的 -32768 或 +32767 个字。可以跳过的实际语句的最大数目取决于程序中使用的语句组合（单字、双字或三字语句）。

（8）JNBI　　当 BR = 0 时跳转

指令格式为：JNBI <跳转标签>

当状态位 BR 为 0 时，JNBI <跳转标签>就中断线性程序扫描，并跳转到一个跳转目标。线性程序扫描在跳转目标处继续执行。由跳转标签确定跳转目标。允许向前跳转和向后跳转。只能在一个块内执行跳转，即跳转指令和跳转目标必须位于同一个块内。跳转目标在该块内必须唯一。最大跳转距离为程序代码的 -32768 或 +32767 个字。可以跳过的实际语句的最大数目取决于程序中使用的语句组合（单字、双字或三字语句）。

（9）JO　　当 OV = 1 时跳转

指令格式为：JO <跳转标签>

当状态位 OV 为 1 时，JO <跳转标签>就中断线性程序扫描，并跳转到一个跳转目标。线性程序扫描在跳转目标处继续执行。由跳转标签确定跳转目标。允许向前跳转和向后跳转。只能在一个块内执行跳转，即跳转指令和跳转目标必须位于同一个块内。跳转目标在该块内必须唯一。最大跳转距离为程序代码的 -32768 或 +32767 个字。可以跳过的实际语句的最大数目取决于程序中使用的语句组合（单字、双字或三字语句）。在一个组合的算术指令中，在每个单独的算术指令后检查是否发生溢出，以确保每个中间结果都位于允许范围内，或使用指令 JOS。

例：

```
L      MW10
L      3
*I                    //将 MW10 的内容乘以“3”
JO     OVER           //当结果超出最大范围（OV=1）时，跳转
T      MW10           //当不执行跳转时，在此继续执行程序扫描
A      M 4.0
R      M 4.0
JU     NEXT
OVER：ANM 4.0         //跳转到跳转标签 OVER 后，在此继续执行程序扫描
S      M 4.0
NEXT：NOP 0           //跳转到跳转标签 NEXT 后，在此继续执行程序扫描
```

（10）JOS　　当 OS = 1 时跳转

指令格式为：JOS <跳转标签>

当状态位 OS 为 1 时，JOS <跳转标签>就中断线性程序扫描，并跳转到一个跳转目标。线性程序扫描在跳转目标处继续执行。由跳转标签确定跳转目标。允许向前跳转和向后跳转。只能在一个块内执行跳转，即跳转指令和跳转目标必须位于同一个块内。跳转目标在该块内必须唯一。最大跳转距离为程序代码的 -32768 或 +32767 个字。可以跳过的实际语句的最大数目取决于程序中使用的语句组合（单字、双字或三字语句）。

例：

```
L      IW10
L      MW12
*I
L      DBW25
+I
L      MW14
-I
JOS    OVER          //在计算 OS=1 期间，如果 3 个指令中有一个发生溢出，则跳转
T      MW16          //当不执行跳转时，在此继续执行程序扫描
A      M4.0
R      M4.0
JU     NEXT
OVER：AN M 4.0   //跳转到跳转标签 OVER 后，在此继续执行程序扫描
S      M 4.0
NEXT：NOP 0       //跳转到跳转标签 NEXT 后，在此继续执行程序扫描
```

注意：在这种情况下，不要使用 JO 指令。当发生溢出时，JO 指令只检查上一条 -I 指令

(11) JZ　　当为零时跳转

指令格式为：JZ <跳转标签>

当状态位 CC 1 = 0 且 CC 0 = 0 时，JZ <跳转标签>（当结果 = 0 时跳转）就中断线性程序扫描，并跳转到一个跳转目标。线性程序扫描在跳转目标处继续执行。由跳转标签确定跳转目标。允许向前跳转和向后跳转。只能在一个块内执行跳转，即跳转指令和跳转目标必须位于同一个块内。跳转目标在该块内必须唯一。最大跳转距离为程序代码的 -32768 或 +32767 个字。可以跳过的实际语句的最大数目取决于程序中使用的语句组合（单字、双字或三字语句）。

例：

```
L      MW10
SRW    1
JZ     ZERO            //当被移出的位 = 0 时，跳转到跳转标签 ZERO
L      MW2             //当不执行跳转时，在此继续执行程序扫描
INC    1
T      MW2
JU     NEXT
ZERO：L MW4        //跳转到跳转标签 ZERO 后，在此继续执行程序扫描
INC    1
T      MW4
NEXT：NOP 0        //跳转到跳转标签 NEXT 后，在此继续执行程序扫描
```

(12) JN　　当不为零时跳转

指令格式为：JN <跳转标签>

当由状态位 CC 1 和 CC 0 指示的结果大于或小于零时（CC 1 = 0/CC 0 = 1 或 CC 1 = 1/CC 0 = 0），JN <跳转标签>（当结果 <> 0 时跳转）就中断线性程序扫描，并跳转到一个跳转目标。线性程序扫描在跳转目标处继续执行。由跳转标签确定跳转目标。允许向前跳转和向后跳转。只

能在一个块内执行跳转，即跳转指令和跳转目标必须位于同一个块内。跳转目标在该块内必须唯一。最大跳转距离为程序代码的 -32768 或 +32767 个字。可以跳过的实际语句的最大数目取决于程序中使用的语句组合（单字、双字或三字语句）。

例：

```
L      IW8
L      MW12
XOW
JN    NOZE          //当 ACCU 1-L 的内容不等于零时跳转
AN    M 4.0         //当不执行跳转时，在此继续执行程序扫描
S     M 4.0
JU    NEXT
NOZE：ANM 4.1      //跳转到跳转标签 NOZE 后，在此继续执行程序扫描
S      M 4.1
NEXT：NOP 0        //跳转到跳转标签 NEXT 后，在此继续执行程序扫描
```

（13）JP 当为正时跳转

指令格式为：JP <跳转标签>

当状态位 CC 1 =1 且 CC 0 = 0 时，JP <跳转标签>（当结果 < 0 时跳转）就中断线性程序扫描，并跳转到一个跳转目标。线性程序扫描在跳转目标处继续执行。由跳转标签确定跳转目标。允许向前跳转和向后跳转。只能在一个块内执行跳转，即跳转指令和跳转目标必须位于同一个块内。跳转目标在该块内必须唯一。最大跳转距离为程序代码的 -32768 或 +32767 个字。可以跳过的实际语句的最大数目取决于程序中使用的语句组合（单字、双字或三字语句）。

例：

```
L      IW8
L      MW12
-I                  //IW8 的内容减去 MW12 的内容
JP     POS          //当结果 > 0（即 ACCU 1 > 0）时跳转
AN    M4.0          //当不执行跳转时，在此继续执行程序扫描
S     M 4.0
JU    NEXT
POS：ANM 4.1    //跳转到跳转标签 POS 后，在此继续执行程序扫描
S      M4.1
NEXT：NOP 0    //跳转到跳转标签 NEXT 后，在此继续执行程序扫描
```

（14）JM 当为负时跳转

指令格式为：JM <跳转标签>

当状态位 CC 1 = 0 且 CC 0 = 1 时，JM <跳转标签>（当结果 < 0 时跳转）就中断线性程序扫描，并跳转到一个跳转目标。线性程序扫描在跳转目标处继续执行。由跳转标签确定跳转目标。允许向前跳转和向后跳转。只能在一个块内执行跳转，即，跳转指令和跳转目标必须位于同一个块内。跳转目标在该块内必须唯一。最大跳转距离为程序代码的 -32768 或 +32767 个字。可以跳过的实际语句的最大数目取决于程序中使用的语句组合（单字、双字或三字语句）。

例：

```
L      IW8
```

```
L    MW12
-I                //IW8 的内容减去 MW12 的内容
JM   NEG          //当结果 < 0（即，ACCU 1 的内容 < 0）时跳转
A N  M 4.0        //当不执行跳转时，在此继续执行程序扫描
S    M 4.0
JU   NEXT
NEG：ANM4.1       //跳转到跳转标签 NEG 后，在此继续执行程序扫描
S    M4.1
NEXT：NOP 0       //跳转到跳转标签 NEXT 后，在此继续执行程序扫描
```

（15）JPZ 当为正或零时跳转

指令格式为：JPZ ＜跳转标签＞

当由状态位 CC 1 和 CC 0 指示的结果大于或等于零时（CC 1 =0/CC 0 =0 或 CC 1 =1/CC 0 =0），JPZ ＜跳转标签＞（当结果 > = 0 时跳转）就中断线性程序扫描，并跳转到一个跳转目标。线性程序扫描在跳转目标处继续执行。由跳转标签确定跳转目标。允许向前跳转和向后跳转。只能在一个块内执行跳转，即，跳转指令和跳转目标必须位于同一个块内。跳转目标在该块内必须唯一。最大跳转距离为程序代码的 -32768 或 +32767 个字。可以跳过的实际语句的最大数目取决于程序中使用的语句组合（单字、双字或三字语句）。

例：

```
L    IW8
L    MW12
-I                //IW8 的内容减去 MW12 的内容
JPZ  REG0         //当结果 > =0（即，ACCU 1 的内容 > = 0）时跳转
AN   M 4.0        //当不执行跳转时，在此继续执行程序扫描
S    M 4.0
JU   NEXT
REG0：ANM 4.1     //跳转到跳转标签 REG0 后，在此继续执行程序扫描
S    M4.1
NEXT：NOP 0       //跳转到跳转标签 NEXT 后，在此继续执行程序扫描
```

（16）JMZ 当为负或零时跳转

指令格式为：JMZ ＜跳转标签＞

当由状态位 CC 1 和 CC 0 指示的结果小于或等于零时（CC 1 =0/CC 0 =0 或 CC 1 =0/CC 0 =1），JMZ ＜跳转标签＞（当结果 < = 0 时跳转）就中断线性程序扫描，并跳转到一个跳转目标。线性程序扫描在跳转目标处继续执行。由跳转标签确定跳转目标。允许向前跳转和向后跳转。只能在一个块内执行跳转，即，跳转指令和跳转目标必须位于同一个块内。跳转目标在该块内必须唯一。最大跳转距离为程序代码的 -32768 或 +32767 个字。可以跳过的实际语句的最大数目取决于程序中使用的语句组合（单字、双字或三字语句）。

例：

```
L    IW8
L    MW12
-I                //IW8 的内容减去 MW12 的内容
JMZ  RGE0         //当结果 < =0 时跳转（即，ACCU 1 的内容 < = 0）
```

```
AN    M4.0          //当不执行跳转时，在此继续执行程序扫描
S     M4.0
JU    NEXT
RGE0：ANM4.1        //跳转到跳转标签 RGE0 后，在此继续执行程序扫描
S     M4.1
NEXT：NOP 0         //跳转到跳转标签 NEXT 后，在此继续执行程序扫描
```

（17）JUO　　当无序时跳转

指令格式为：JUO <跳转标签>

当状态位 CC 1 = 1 且 CC 0 = 1 时，JUO <跳转标签>就中断线性程序扫描，并跳转到一个跳转目标。线性程序扫描在跳转目标处继续执行。由跳转标签确定跳转目标。允许向前跳转和向后跳转。只能在一个块内执行跳转，即跳转指令和跳转目标必须位于同一个块内。跳转目标在该块内必须唯一。最大跳转距离为程序代码的 -32768 或 +32767 个字。可以跳过的实际语句的最大数目取决于程序中使用的语句组合（单字、双字或三字语句）。

在三种情况下，状态位 CC 1 = 1 且 CC 0 = 1：发生被零除时、使用了非法指令时、浮点比较的结果为“无序”（即使用了一种无效格式时）。

例：

```
L     MD10
L     ID2
/D                  //MD10 的内容除以 ID2 的内容
JUO   ERRO          //当被零除时跳转（即，ID2 = 0）
T     MD14          //当不执行跳转时，在此继续执行程序扫描
A     M4.0
R     M4.0
JU    NEXT
ERRO：ANM4.0        //跳转到跳转标签 ERRO 后，在此继续执行程序扫描
SM 4.0
NEXT：NOP 0         //跳转到跳转标签 NEXT 后，在此继续执行程序扫描
```

（18）LOOP　　循环

指令格式为：LOOP <跳转标签>

LOOP <跳转标签>（对 ACCU 1-L 进行减 1 操作，并在 ACCU 1-L < > 0 时跳转）可简化循环编程。ACCU 1 - L 中包含循环计数器。指令跳转到指定的跳转目标。只要 ACCU 1-L 的内容不等于 0，就一直执行跳转。在跳转目标处继续执行线性程序扫描。跳转目标由跳转标签确定。允许向前跳转和向后跳转。只能在一个块内执行跳转，即跳转指令和跳转目标必须位于同一个块内。跳转目标在该块内必须唯一。最大跳转距离为程序代码的 -32768 或 +32767 个字。可以跳过的实际语句的最大数目取决于程序中使用的语句组合（单字、双字或三字语句）。

例（计算因子为 5）：

```
L     l#1           //将整型常数（32 位）装载到 ACCU 1 中
T     MD20          //将 ACCU1 的内容传送给 MD20（初始化）
L     5             //将循环周期的数目装载到 ACCU 1-L 中
NEXT：TMW10         //跳转标签 = 循环开始/将 ACCU 1-L 传送给循环计数器
L     MD20
```

```
*D//MD20 的当前内容乘以 MB10 的当前内容
T       MD20     //将相乘结果传送给 MD20
L       MW10     //将循环计数器的内容装载到 ACCU 1 中
LOOP NEXT        //对 ACCU 1 的内容进行减 1 操作，当 ACCU 1-L ＞ 0 时，跳转到 NEXT
L       MW24     //完成循环后，在此继续执行程序扫描
L       200
>I
```

2. 梯形图（LAD）的逻辑控制指令

（1）---（JMP）无条件跳转

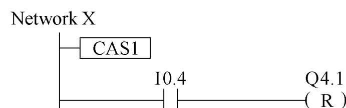
指令符号为：<标号名称>

---（JMP）

---（JMP）（为 1 时在块内跳转）当左侧电源轨道与指令间没有其他梯形图元素时执行的是绝对跳转。每个---（JMP）还都必须有与之对应的目标（LABEL）。跳转指令和标号间的所有指令都不予执行。始终执行跳转，并忽略跳转指令和跳转标号间的指令。

例：





（2）---（JMP）条件跳转

指令符号为：<标号名称>

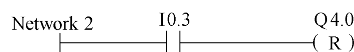
---（JMP）

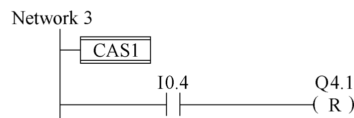
---（JMP）（为 1 时在块内跳转）当前一逻辑运算的 RLO 为“1”时执行的是条件跳转。每个---（JMP）都还必须有与之对应的目标（LABEL）。跳转指令和标号间的所有指令都不予执行。

如果未执行条件跳转，RLO 将在执行跳转指令后变为“1”。

例：







如果 I0.0 ＝“1”，则执行跳转到标号 CAS1。由于该跳转的存在，即使 I0.3 处有逻辑“1”，也不会执行复位输出 Q4.0 的指令。

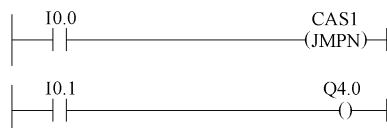
（3）---（JMPN）若“否”则跳转

指令符号为：<标号名称>

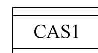
---（JMPN）

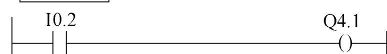
---（JMPN）（若“否”则跳转）相当于在 RLO 为“0”时执行的“转到标号”功能。每一个---（JMPN）都还必须有与之对应的目标（LABEL）。跳转指令和标号间的所有指令都不予执行。如果未执行条件跳转，RLO 将在跳转指令执行后变为“1”。

例：





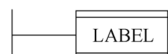




如果 I0.0 ＝“0”，则执行跳转到标号 CAS1。由于存在该跳转，即使 I0.1 处有逻辑“1”，也不会执行复位输出 Q4.0 的指令。

（4）LABEL 标号

指令符号为：



LABEL 是跳转指令目标的标识符。第一个字符必须是字母表中的字母；其他字符可以是字母或数字（例如，CAS1）。每个---（JMP）或---（JMPN）都还必须有与之对应的跳转标号（LABEL）。

七、整型数学运算指令

算术指令组合累加器 1 和累加器 2 的内容，结果存储在累加器 1 中。对于带 2 个累加器的 CPU 而言，累加器 2 的内容保持不变。对于带 4 个累加器的 CPU，将累加器 3 的内容复制到累加器 2 中，将累加器 4 的内容复制到累加器 3 中。累加器 4 中较早的内容保持不变。

使用整数算术，可以对两个整数（16 位和 32 位）执行运算。整数算术指令影响状态字中的下列位：CC1 和 CC0、OV 和 OS。

1. 语句表（STL）的整型数学运算指令

（1） +I　　ACCU 1 ＋ ACCU 2，整型（16 位）

指令格式为：+I

+I（16 位整数相加）将 ACCU 1-L 的内容与 ACCU 2-L 中的内容相加，并将结果存储在 ACCU 1-L 中。将 ACCU 1-L 和 ACCU 2-L 的内容解释为 16 位整数。执行该指令，与 RLO 无关，也不影响 RLO。作为指令运算结果的一个功能，将对状态字的位 CC 1、CC 0、OS 和 OV 进行设置。在发生溢出/下溢时，该指令生成一个 16 位整数，而不是一个 32 位整数。对于带 2 个累加器的 CPU 而言，累加器 2 的结果保持不变。对于带 4 个累加器的 CPU，还将累加器 3 的内容复制到累加器 2 中，将累加器 4 的内容复制到累加器 3 中。累加器 4 的内容保持不变。

例 ：

```
L     IW10          //将 IW10 的数值装载到 ACCU 1-L 中
L     MW14          //将 ACCU 1-L 中的数送至 ACCU 2-L，MW14 的数值送至 ACCU 1-L 中
+I                  //ACCU 2-L 和 ACCU 1-L 相加，结果存储在 ACCU 1-L 中
T     DB1. DBW25    //将 ACCU 1-L（结果）的内容传送到 DB1 的 DBW25 中
```

（2） －I　　ACCU 2-ACCU 1，整型（16 位）

指令格式为：－I

－I（16 位整数相减）从 ACCU 2-L 的内容中减去 ACCU 1-L 的内容，并将结果存储在 ACCU 1-L 中。将 ACCU 1-L 和 ACCU 2-L 的内容解释为 16 位整数。执行该指令，与 RLO 无关，也不影响 RLO。作为指令运算结果的一个功能，将对状态字的位 CC 1、CC 0、OS 和 OV 进行设置。在发生溢出/下溢时，该指令生成一个 16 位整数，而不是一个 32 位整数。对于带 2 个累加器的 CPU 而言，累加器 2 的结果保持不变。对于带 4 个累加器的 CPU，还将累加器 3 的内容复制到累加器 2 中，将累加器 4 的内容复制到累加器 3 中。累加器 4 的内容保持不变。

例：

```
L     IW10          //将 IW10 的数值装载到 ACCU 1-L 中
L     MW14          //将 ACCU 1-L 中的数送至 ACCU 2-L 中，MW14 的数值送至 ACCU 1-L 中
-I                  //从 ACCU 2-L 中减去 ACCU 1-L，结果存储在 ACCU 1-L 中
T     DB1. DBW25    //将 ACCU 1-L（结果）的内容传送到 DB1 的 DBW25 中
```

（3） *I　　ACCU1 ＊ ACCU2，整型（16 位）

指令格式为：*I

*I（乘以 16 位整数）将 ACCU 2-L 的内容乘以 ACCU 1-L 的内容。将 ACCU 1-L 和 ACCU 2-L 的内容解释为 16 位整数。结果作为一个 32 位整数存储在 ACCU 1 中。当状态字的位为 OV1 ＝ 1 和 OS ＝ 1 时，表示结果超出 16 位整数范围。执行该指令，与 RLO 无关，也不影响 RLO。作为指令运算结果的一个功能，将对状态字的位 CC 1、CC 0、OS 和 OV 进行设置。对于带 2 个累加器的 CPU 而言，累加器 2 的结果保持不变。对于带 4 个累加器的 CPU，还将累加器 3 的内容复制到累加器 2 中，将累加器 4 的内容复制到累加器 3 中。

例：

```
L    IW10       //将 IW10 的数值装载到 ACCU 1-L 中
L    MW14       //将 ACCU 1-L 中的数送至 ACCU 2-L 中，MW14 的数值送至 ACCU 1-L 中
*I              //ACCU 2-L 和 ACCU 1-L 相乘，结果存储在 ACCU 1 中
T    DB1.DBD25  //将 ACCU 1（结果）的内容传送到 DB1 的 DBD25 中
```

（4）/I　　ACCU 2 / ACCU 1，整型（16 位）

指令格式为：/I

/I（16 位整数相除）将 ACCU 2-L 的内容除以 ACCU 1-L 的内容。将 ACCU 1-L 和 ACCU 2-L 的内容解释为 16 位整数，结果存储在 ACCU 1 中，包含两个 16 位整数，即商和余数。在 ACCU 1-L 中存储商，在 ACCU 1-H 中存储余数。执行该指令，与 RLO 无关，也不影响 RLO。作为指令运算结果的一个功能，将对状态字的位 CC 1、CC 0、OS 和 OV 进行设置。对于带 2 个累加器的 CPU 而言，累加器 2 的结果保持不变。对于带 4 个累加器的 CPU，还将累加器 3 的内容复制到累加器 2 中，将累加器 4 的内容复制到累加器 3 中。累加器 4 的内容保持不变。

例：

```
L    IW10       //将 IW10 的数值装载到 ACCU 1-L 中
L    MW14       //将 ACCU 1-L 中的数送入 ACCU 2-L 中，MW14 的数值装入 ACCU 1-L 中
/I              //ACCU 2-L 除以 ACCU 1-L；结果存在 ACCU 1 中，L：商、H：余数
T    MD20       //将 ACCU 1（结果）的内容传送到 MD20 中，
```

例：13 除以 4

执行指令（IW10）前，ACCU 2-L 的内容："13"

执行指令（MW14）前，ACCU 1-L 的内容："4"

指令 /I（ACCU 2-L / ACCU 1-L）："13/4"

执行指令后，ACCU 1-L 的内容（商）："3"

执行指令后，ACCU 1-H 的内容（余数）："1"

（5）+　　整型常数（16 位、32 位）

指令格式为：+ <整型常数>

+ <整型常数>将整型常数加到 ACCU 1 的内容中，并将结果存储在 ACCU 1 中。该指令的执行与状态字的位无关，也不影响状态字的位。对于带 2 个累加器的 CPU 而言，累加器 2 的结果保持不变。对于带 4 个累加器的 CPU，还将累加器 3 的内容复制到累加器 2 中，将累加器 4 的内容复制到累加器 3 中。累加器 4 的内容保持不变。+ <16 位整型常数>：将一个 16 位整型常数（范围为 -32768 ~ +32767）加到 ACCU 1-L 的内容中，然后将结果存储在 ACCU 1-L 中。对于带 2 个累加器的 CPU 而言，累加器 2 的结果保持不变。对于带 4 个累加器的 CPU，还将累加器 3 的内容复制到累加器 2 中，将累加器 4 的内容复制到累加器 3 中。累加器 4 的内容保持不变。+ <32 位整型常数>：将一个 32 位整型常数（范围为 -2，147，483，648 ~ 2，147，483，647）加到 ACCU 1 的内容中，然后将结果存储在 ACCU 1 中。对于带 2 个累加器的 CPU 而言，累加器 2 的结果保持不变。对于带 4 个累加器的 CPU，还将累加器 3 的内容复制到累加器 2 中，将累加器 4 的内容复制到累加器 3 中。累加器 4 的内容保持不变。

例 1：

```
L    IW10       //将 IW10 的数值装载到 ACCU 1-L 中
L    MW14       //将 ACCU 1-L 中的数送至 ACCU 2-L 中，MW14 的数值送至 ACCU 1-L 中
+I              //ACCU 2-L 和 ACCU 1-L 相加，结果存储在 ACCU 1-L 中
```

```
+      25              //ACCU 1-L 与 25 相加，将结果存储在 ACCU 1-L 中
T      DB1. DBW25      //将 ACCU 1-L（结果）的内容传送到 DB1 的 DBW25 中
```

例 2：

```
L      IW12
L      IW14
+      100          //ACCU 1-L 和 100 相加，结果存储在 ACCU 1-L 中
>I                  //当 ACCU 2 > ACCU 1 或 IW12 >（IW14 + 100）时
JC     NEXT         //则条件跳转到跳转标签 NEXT
```

例 3：

```
L      MD20
L      MD24
+D                  //ACCU 1 和 ACCU 2 相加，结果存储在 ACCU 1 中
+      l# -200      //ACCU 1 和 -200 相加，结果存储在 ACCU 1 中
T      MD28
```

（6）+D　ACCU 1 + ACCU 2，长整型（32 位）

指令格式为：+D

+D（32 位整数相加）将 ACCU 1 的内容与 ACCU 2 中的内容相加，并将结果存储在 ACCU 1 中。将 ACCU 1 和 ACCU 2 的内容解释为 32 位整数。执行该指令，与 RLO 无关，也不影响 RLO。作为指令运算结果的一个功能，将对状态字的位 CC 1、CC 0、OS 和 OV 进行设置。对于带 2 个累加器的 CPU 而言，累加器 2 的结果保持不变。对于带 4 个累加器的 CPU，还将累加器 3 的内容复制到累加器 2 中，将累加器 4 的内容复制到累加器 3 中。累加器 4 的内容保持不变。

例：

```
L      ID10         //将 ID10 的数值装载到 ACCU 1 中
L      MD14         //将 ACCU 1 的内容装载到 ACCU 2 中，将 MD14 的数值送入 ACCU 1 中
+D                  //ACCU 2 和 ACCU 1 相加，结果存储在 ACCU 1 中
T      DB1. DBD25   //将 ACCU 1（结果）的内容传送到 DB1 的 DBD25 中
```

（7）-D　ACCU 2 - ACCU 1，长整型（32 位）

指令格式为：-D

-D（32 位整数相减）从 ACCU 2 的内容中减去 ACCU 1 的内容，并将结果存储在 ACCU 1 中。将 ACCU 1 和 ACCU 2 的内容解释为 32 位整数。执行该指令，与 RLO 无关，也不影响 RLO。作为指令运算结果的一个功能，将对状态字的位 CC 1、CC 0、OS 和 OV 进行设置。对于带 2 个累加器的 CPU 而言，累加器 2 的结果保持不变。对于带 4 个累加器的 CPU，还将累加器 3 的内容复制到累加器 2 中，将累加器 4 的内容复制到累加器 3 中。累加器 4 的内容保持不变。

例：

```
L      ID10         //将 ID10 的数值装载到 ACCU 1 中
L      MD14         //将 ACCU 1 的内容装载到 ACCU 2 中，将 MD14 的数值送入 ACCU 1 中
-D                  //从 ACCU 2 中减去 ACCU 1，结果存储在 ACCU 1 中
T      DB1. DBD25   //将 ACCU 1（结果）的内容传送到 DB1 的 DBD25 中
```

（8）*D　ACCU 1 * ACCU 2，长整型（32 位）

指令格式为：*D

*D（乘以 32 位整数）将 ACCU 2 的内容乘以 ACCU 1 的内容。将 ACCU 1 的内容和 ACCU 2

的内容解释为 32 位整数。结果作为一个 32 位整数存储在 ACCU 1 中。当状态字的位为 OV1 = 1 和 OS = 1 时，表示结果超出 32 位整数范围。执行该指令，与 RLO 无关，也不影响 RLO。作为指令运算结果的一个功能，将对状态字的位 CC 1、CC 0、OS 和 OV 进行设置。对于带 2 个累加器的 CPU 而言，累加器 2 的内容保持不变。对于带 4 个累加器的 CPU，还将累加器 3 的内容复制到累加器 2 中，将累加器 4 的内容复制到累加器 3 中。累加器 4 的内容保持不变。

例：

```
L     ID10          //将 ID10 的数值装载到 ACCU 1 中
L     MD14          //将 ACCU 1 的内容装载到 ACCU 2 中，将 MD14 的内容送入 ACCU 1 中
*D                  //ACCU 2 和 ACCU 1 相乘，结果存储在 ACCU 1 中
T     DB1.DBD25     //将 ACCU 1（结果）的内容传送到 DB1 的 DBD25 中
```

（9）/D　ACCU 2 / ACCU 1，长整型（32 位）

指令格式为：/D

/D（32 位整数相除）将 ACCU 2 的内容除以 ACCU 1。将 ACCU 1 和 ACCU 2 的内容解释为 32 位整数。在 ACCU 1 中存储该指令的结果。结果只给出商，不给出余数（指令 MOD 可用于获取余数）。执行该指令，与 RLO 无关，也不影响 RLO。作为指令运算结果的一个功能，将对状态字的位 CC 1、CC 0、OS 和 OV 进行设置。对于带 2 个累加器的 CPU 而言，累加器 2 的结果保持不变。对于带 4 个累加器的 CPU，还将累加器 3 的内容复制到累加器 2 中，将累加器 4 的内容复制到累加器 3 中。累加器 4 的内容保持不变。

例：

```
L     ID10     //将 ID10 的数值装载到 ACCU 1 中
L     MD14     //将 ACCU 1 的内容装载到 ACCU 2 中，将 MD14 的数值送入 ACCU 1 中
/D             //ACCU 2 除以 ACCU 1，结果存储在 ACCU 1 中（商）
T     MD20     //将 ACCU 1（结果）的内容传送到 MD20 中
```

例：13 除以 4

执行指令（ID10）前，ACCU 2 的内容："13"

执行指令（MD14）前，ACCU 1 的内容："4"

指令/D（ACCU 2 / ACCU 1）："13/4"

执行指令后，ACCU 1 的内容（商）："3"

（10）MOD　除法余数，长整型（32 位）

指令格式为：MOD

MOD（32 位整数除法的余数）将 ACCU 2 的内容除以 ACCU 1 的内容。将 ACCU 1 的内容和 ACCU 2 的内容解释为 32 位整数。在 ACCU 1 中存储该指令的结果。结果只给出除法的余数，不给出商（指令/D 可用于获取商）。执行该指令，与 RLO 无关，也不影响 RLO。作为指令运算结果的一个功能，将对状态字的位 CC 1、CC 0、OS 和 OV 进行设置。对于带 2 个累加器的 CPU 而言，累加器 2 的结果保持不变。对于带 4 个累加器的 CPU，还将累加器 3 的内容复制到累加器 2 中，将累加器 4 的内容复制到累加器 3 中。累加器 4 的内容保持不变。

例：

```
L     ID10     //将 ID10 的数值装载到 ACCU 1 中
L     MD14     //将 ACCU 1 的内容装载到 ACCU 2 中。将 MD14 的数值送到 ACCU 1 中
MOD            //ACCU 2 除以 ACCU 1，结果存储在 ACCU 1 中（余数）
T     MD20     //将 ACCU 1（结果）的内容传送到 MD20 中
```

例：13 除以 4

执行指令（ID10）前，ACCU 2 的内容：″13″

执行指令（MD14）前，ACCU 1 的内容：″4″

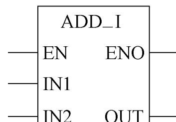
指令 MOD（ACCU 2 / ACCU 1）：　　″13/4″

执行指令后，ACCU 1 的内容（余数）：″1″

2. 梯形图（LAD）的整型数学运算指令

（1）ADD_ I　整数加

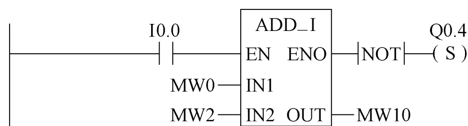
指令符号为：



EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN1（被加数）和 IN2（加数）的数据类型为 INT 型，存储区为 I、Q、M、L、D 或常数。OUT（加法结果）的数据类型为 INT 型，存储区为 I、Q、M、L、D。在起用（EN）输入端通过一个逻辑“1”来激活 ADD_ I（整数加）。IN1 和 IN2 相加，结果通过 OUT 查看。如果该结果超出了整数（16 位）允许的范围，OV 位和 OS 位将为“1”并且 ENO 为逻辑“0”，这样便不执行此数学框后由 ENO 连接的其他函数（层叠排列）。

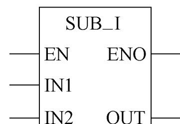
例：

如果 I0. 0 ＝“1”，则激活 ADD_ I 框。MW0 ＋ MW2 相加的结果输出到 MW10。如果结果超出整数的允许范围，则设置输出 Q4. 0。



（2）SUB_ I　整数减

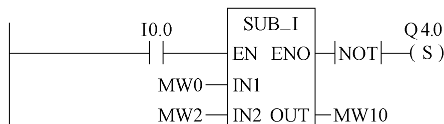
指令符号为：



EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN1（被减数）和 IN2（减数）的数据类型为 INT 型，存储区为 I、Q、M、L、D 或常数。OUT（减法结果）的数据类型为 INT 型，存储区为 I、Q、M、L、D。在起用（EN）输入端通过逻辑“1”激活 SUB_ I（整数减）。IN1 减去 IN2，结果可通过 OUT 查看。如果该结果超出了整数（16 位）允许的范围，OV 位和 OS 位将为“1”并且 ENO 为逻辑“0”，这样便不执行此数学框后由 ENO 连接的其他函数（层叠排列）。

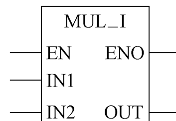
例：

如果 I0. 0 ＝“1”，则激活 SUB_ I 框。MW0 － MW2 相减的结果输出到 MW10。如果结果超出整数允许范围，或者 I0. 0 信号状态 ＝ 0，则设置输出 Q4. 0。



（3）MUL_ I　整数乘

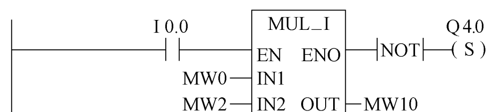
指令符号为：



EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN1（被乘数）和 IN2（乘数）的数据类型为 INT 型，存储区为 I、Q、M、L、D 或常数。OUT（运算结果）的数据类型为 INT 型，存储区为 I、Q、M、L、D。在起用（EN）输入端通过逻辑“1”激活 MUL_ I（整数乘）。IN1 和 IN2 相乘，结果通过 OUT 查看。如果该结果超出了整数（16 位）允许的范围，OV 位和 OS 位将为“1”并且 ENO 为逻辑“0”，这样便不执行此数学框后由 ENO 连接的其他函数（层叠排列）。

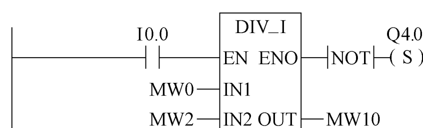
例：

如果 I0.0 = "1"，则激活 MUL_ I 框。MW0 × MW2 相乘的结果输出到 MD10。如果结果超出整数的允许范围，则设置输出 Q4.0。



（4）DIV_ I　整数除

指令符号为：



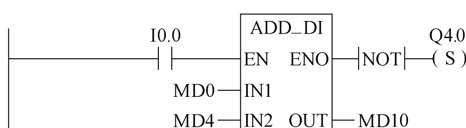
EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN1（被除数）和 IN2（除数）的数据类型为 INT 型，存储区为 I、Q、M、L、D 或常数。OUT（运算结果）的数据类型为 INT 型，存储区为 I、Q、M、L、D。在起用（EN）输入端通过逻辑"1"激活 DIV_ I（整数除）。IN1 除以 IN2，结果可通过 OUT 查看。如果该结果超出了整数（16 位）允许的范围，OV 位和 OS 位将为"1"并且 ENO 为逻辑"0"，这样便不执行此数学框后由 ENO 连接的其 3 上函数（层叠排列）。

例：

如果 I0.0 = "1"，则激活 DIV_ I 框。MW0 除以 MW2 的结果输出到 MW10。如果结果超出整数的允许范围，则设置输出 Q4.0。

（5）ADD_ DI　长整数加

指令符号为：



EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN1（被加数）和 IN2（加数）的数据类型为 DINT 型，存储区为 I、Q、M、L、D 或常数。OUT（运算结果）的数据类型为 DINT 型，存储区为 I、Q、M、L、D。在起用（EN）输入端通过逻辑"1"激活 ADD_ DI（长整数加）。IN1 和 IN2 相加，结果通过 OUT 查看。如果该结果超出了长整数（32 位）允许的范围，OV 位和 OS 位将为"1"并且 ENO 为逻辑"0"，这样便不执行此数学框后由 ENO 连接的其他函数（层叠排列）。

例：

如果 I0.0 = "1"，则激活 ADD_ DI 框。MD0 + MD4 相加的结果输出到 MD10。如果结果超出长整数的允许范围，则设置输出 Q4.0。

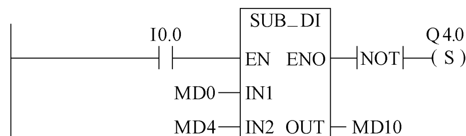
（6）SUB_ DI　长整数减

指令符号为：

EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN1（被减数）和 IN2（减数）的数据类型为 DINT 型，存储区为 I、Q、M、L、D 或常数。OUT（运算结果）的数据类型为 DINT 型，存储区为 I、Q、M、L、D。在起用（EN）输入端通过逻辑"1"激活 SUB_ DI（长整数减）。IN1 减去 IN2，结果可通过 OUT 查看。如果该结果超出了长整数（32 位）允许的范围，OV 位和 OS 位将为"1"并且 ENO 为逻辑"0"，这样便不执行此数学框后由 ENO 连接的其他函数（层叠排列）。

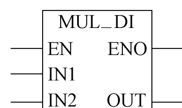
例：

如果 I0.0 = "1"，则激活 SUB_ DI 框。MD0 − MD4 相减的结果输出到 MD10。如果结果超出长整数的允许范围，则设置输出 Q4.0。



（7）MUL_ DI　长整数乘

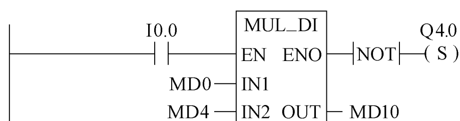
指令符号为：



EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN1（被乘数）和 IN2（乘数）的数据类型为 DINT 型，存储区为 I、Q、M、L、D 或常数。OUT（运算结果）的数据类型为 DINT 型，存储区为 I、Q、M、L、D。在起用（EN）输入端通过逻辑“1”激活 MUL_ DI（长整数乘）。IN1 和 IN2 相乘，结果通过 OUT 查看。如果该结果超出了长整数（32 位）允许的范围，OV 位和 OS 位将为“1”并且 ENO 为逻辑“0”，这样便不执行此数学框后由 ENO 连接的其他函数（层叠排列）。

例：

如果 I0. 0 = “1”，则激活 MUL_ DI 框。MD0 × MD4 相乘的结果输出到 MD10。如果结果超出长整数的允许范围，则设置输出 Q4. 0。



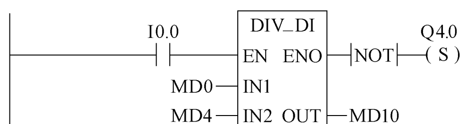
（8）DIV_ DI　长整数除

指令符号为：

EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN1（被除数）和 IN2（除数）的数据类型为 DINT 型，存储区为 I、Q、M、L、D 或常数。OUT（运算结果）的数据类型为 DINT 型，存储区为 I、Q、M、L、D。在起用（EN）输入端通过逻辑“1”激活 DIV_ DI（长整数除）。IN1 除以 IN2，结果可通过 OUT 查看。长整型除法不产生余数。如果该结果超出了长整数（32 位）允许的范围，OV 位和 OS 位将为“1”并且 ENO 为逻辑“0”，这样便不执行此数学框后由 ENO 连接的其他函数（层叠排列）。

例：

如果 I0. 0 = “1”，则激活 DIV_ DI 框。MD0: MD4 的相除结果输出到 MD10。如果结果超出长整数的允许范围，则设置输出 Q4. 0。



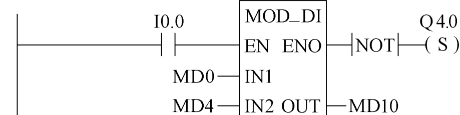
（9）MOD_ DI　返回长整数余数

指令符号为：

EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN1（被除数）和 IN2（除数）的数据类型为 DINT 型，存储区为 I、Q、M、L、D 或常数。OUT（除运算的余数）的数据类型为 DINT 型，存储区为 I、Q、M、L、D。在起用（EN）输入端通过逻辑“1”激活 MOD_ DI（返回长整数余数）。IN1 除以 IN2，余数可通过 OUT 查看。如果该结果超出了长整数（32 位）允许的范围，OV 位和 OS 位将为“1”并且 ENO 为逻辑“0”，这样便不执行此数学框后由 ENO 连接的其他函数（层叠排列）。

例：

如果 I0. 0 = “1”，则激活 DIV_ DI 框。MD0: MD4 相除的余数输出到 MD10。如果余数超出长整数的允许范围，则设置输出 Q4. 0。



八、浮点运算指令

数学运算指令会组合累加器 1 和累加器 2 的内容，并将结果存储在累加器 1 中。对于具有两个累加器的 CPU，累加器 2 的内容保持不变。对于具有 4 个累加器的 CPU，则会将累加器 3 的内容复制到累加器 2 中，并将累加器 4 的内容复制到累加器 3 中。累加器 4 的旧内容保持不变。

IEEE 32 位浮点数属于称作实数（REAL）的数据类型。可使用浮点运算指令通过两个 32 位 IEEE 浮点数来执行数学运算指令。基本运算类型会影响状态字中的 CC 1 和 CC 0、OV 和 OS 位。

1. 语句表（STL）的浮点运算指令

（1）+R　　将 ACCU 1 和 ACCU 2 作为浮点数（32 位 IEEE-FP）相加

指令格式为：+R

+R（加 32 位 IEEE 浮点数）将累加器 1 与累加器 2 的内容相加，并将结果存储到累加器 1 中。将累加器 1 和累加器 2 的内容解释为 32 位 IEEE 浮点数。该指令的执行也不影响 RLO。结果会对状态位 CC 1、CC 0、OS 和 OV 进行设置。对于具有两个累加器的 CPU，累加器 2 的内容保持不变。对于具有 4 个累加器的 CPU，则会将累加器 3 的内容复制到累加器 2，并将累加器 4 的内容复制到累加器 3。累加器 4 的内容保持不变。

例：

```
OPN DB10
L     ID10     //将 ID10 的值装载到 ACCU 1 中
L     MD14     //将 ACCU 1 的值装载到 ACCU 2 中，将 MD14 的值装载到 ACCU 1 中
+R             //ACCU 2 和 ACCU 1 相加，结果存储到 ACCU 1 中
T     DBD25    //将 ACCU 1 的内容（结果）传送到 DB10 的 DBD25 中
```

（2）-R　　将 ACCU 2 与 ACCU 1 作为浮点数（32 位 IEEE-FP）相减

指令格式为：-R

-R（减 32 位 IEEE 浮点数）从累加器 2 减去累加器 1 的内容，并将结果存储到累加器 1 中。将累加器 1 和累加器 2 的内容解释为 32 位 IEEE 浮点数。结果存储在累加器 1 中。该指令的执行既不考虑也不影响 RLO。结果会对状态位 CC 1、CC 0、OS 和 OV 进行设置。

对于具有两个累加器的 CPU，累加器 2 的内容保持不变。对于具有 4 个累加器的 CPU，则会将累加器 3 的内容复制到累加器 2，并将累加器 4 的内容复制到累加器 3。累加器 4 的内容保持不变。

例：

```
OPN DB10
L     ID10     //将 ID10 的值装载到 ACCU 1 中
L     MD14     //将 ACCU 1 的值装载到 ACCU 2 中，将 MD14 的值装载到 ACCU 1 中
-R             //ACCU 2 减 ACCU 1，结果存储到 ACCU 1 中
T     DBD25    //将 ACCU 1 的内容（结果）传送到 DB10 的 DBD25 中
```

（3）*R　　将 ACCU 1 和 ACCU 2 作为浮点数（32 位 IEEE-FP）相乘

指令格式为：*R

*R（乘 32 位 IEEE 浮点数）将累加器 2 与累加器 1 的内容相乘。将累加器 1 和累加器 2 中的内容解释为 32 位 IEEE 浮点数，结果作为 32 位 IEEE 浮点数存储在累加器 1 中。该指令的执行既不考虑也不影响 RLO。结果会对状态位 CC 1、CC 0、OS 和 OV 进行设置。对于具有两个累加器的 CPU，累加器 2 的内容保持不变。对于具有 4 个累加器的 CPU，则会将累加器 3 的内容复制到累加器 2，并将累加器 4 的内容复制到累加器 3。累加器 4 的内容保持不变。

例

```
OPN     DB10
L     ID10     //将 ID10 的值装载到 ACCU 1 中
L     MD14     //将 ACCU 1 的值装载到 ACCU 2 中，将 MD14 的值装载到 ACCU 1 中
*R             //ACCU 2 和 ACCU 1 相乘，结果存储到 ACCU 1 中
T     DBD25    //将 ACCU 1 的内容（结果）传送到 DB10 的 DBD25 中
```

（4）/R　　将 ACCU 2 与 ACCU 1 作为浮点数（32 位 IEEE-FP）相除

指令格式为：/R

/R（除 32 位 IEEE 浮点数）将累加器 2 的内容除以累加器 1 的内容。将累加器 1 和累加器 2 的内容解释为 32 位 IEEE 浮点数。该指令的执行既不考虑也不影响 RLO。结果会对状态位 CC1、CC 0、OS 和 OV 进行设置。对于具有两个累加器的 CPU，累加器 2 的内容保持不变。对于具有 4 个累加器的 CPU，则会将累加器 3 的内宾复制到累加器 2，并将累加器 4 的内容复制到累加器 3。

例：

```
OPN     DB10
L       ID10     //将 ID10 的值装载到 ACCU 1 中
L       MD14     //将 ACCU 1 的内容装载到 ACCU 2 中，将 MD14 的值送至 ACCU 1 中
/R               //ACCU 2 除以 ACCU 1，结果存储到 ACCU 1 中
T       DBD20    //将 ACCU 1 的内容（结果）传送到 DB10 的 DBD20 中
```

（5）ABS　浮点数的（32 位 IEEE -FP）绝对值

指令格式为：ABS

ABS（32 位 IEEE -FP 的绝对值）在 ACCU 1 中计算浮点数（32 位 IEEE 浮点数）的绝对值，结果存储在累加器 1 中。该指令的执行既不考虑也不影响状态位。

例：

```
L       ID8      //将值装载到 ACCU 1 中（实例：ID8 = -1.5E+02）
ABS              //生成绝对值；结果存储在 ACCU 1 中
T       MD10     //将结果传送到 MD10（实例：结果 = 1.5E+02）
```

（6）SQR　计算浮点数（32 位）的二次方

指令格式为：SQR

SQR（计算 IEEE -FP 32 位浮点数的二次方）在累加器 1 中计算浮点数（32 位，IEEE -FP）的二次方，结果存储在累加器 1 中。此指令影响 CC 1、CC 0、OV 和 OS 状态字位。累加器 2 的内容（以及具有 4 个累加器的 CPU 的累加器 3 和累加器 4 的内容）保持不变。

例：

```
OPN     DB17     //打开数据块 DB17
L       DBD0     //将数据双字 DBD0 的值装载到 ACCU 1 中（此值必须为浮点数格式）
SQR              //在 ACCU 1 中计算浮点数（32 位，IEEE -FP）的二次方，将结果存
                   ACCU 1 中
AN      OV       //扫描状态字中的 OV 位，确定是否是“0”
JC      OK       //如果在执行 SQR 指令期间未出错，则跳转到 OK 跳转标签
BEU              //如果执行 SQR 指令时出错，则块无条件结束
OK：T   DBD4     //将结果从 ACCU 1 传送到数据块中的双字 DBD4
```

（7）SQRT　计算浮点数（32 位）的平方根

指令格式为：SQRT

SQRT（计算 32 位 IEEE -FP 浮点数的平方根）在累加器 1 中计算浮点数（32 位，IEEE -FP）的平方根，结果存储在累加器 1 中。输入值必须大于或等于零。然后得出结果也是正数。唯一的例外是 -0 的平方根是 -0。此指令影响 CC 1、CC 0、OV 和 OS 状态字位。累加器 2 的内容（以及具有 4 个累加器的 CPU 的累加器 3 和累加器 4 的内容）保持不变。

例：

```
L       MD10     //将存储器双字 MD10 的值装载到 ACCU 1 中（此值必须为浮点数格式）
SQRT             //在 ACCU 1 中计算浮点数的平方根，将结果存储在 ACCU 1 中
```

```
AN      OV      //扫描状态字中的 OV 位，确定是否是“0”
JC      OK      //如果在执行 SQRT 指令期间未出错，则跳转到 OK 跳转标签
BEU             //如果执行 SQRT 指令时出错，则块无条件结束
OK：T   MD20    //将结果从 ACCU 1 传送到存储器双字 MD20
```

（8）EXP　计算浮点数（32 位）的指数值

指令格式为：EXP

EXP（计算 32 位 IEEE -FP 浮点数的指数值）在累加器 1 中计算浮点数（32 位，IEEE -FP）的指数值（底数为 e 的指数值），结果存储在累加器 1 中。此指令影响 CC 1、CC 0、OV 和 OS 状态字位。累加器 2 的内容（以及具有 4 个累加器的 CPU 的累加器 3 和累加器 4 的内容）保持不变。

例：

```
L       MD10    //将存储器双字 MD10 的值（此值必须为浮点数格式）送入 ACCU 1 中
EXP             //在 ACCU 1 中计算浮点数的指数值，底数为 e，将结果存在 ACCU 1 中
AN      OV      //扫描状态字中的 OV 位，确定是否是“0”
JC      OK      //如果在执行 EXP 指令期间未出错，则跳转到 OK 跳转标签
BEU             //如果执行 EXP 指令时出错，则块无条件结束
OK：T   MD20    //将结果从 ACCU 1 传送到存储器双字 MD20
```

（9）LN　计算浮点数（32 位）的自然对数

指令格式为：LN

LN（计算 IEEE -FP 32 位浮点数的自然对数）在累加器 1 中计算浮点数（32 位，IEEE -FP）的自然对数（底数为 e 的对数），结果存储在累加器 1 中。输入值必须大于或等于零。此指令影响 CC 1、CC 0、UO 和 OV 状态字位。累加器 2 的内容（以及具有 4 个累加器的 CPU 的累加器 3 和累加器 4 的内容）保持不变。

例：

```
L       MD10    //将存储器双字 MD10 的值（此值必须为浮点数格式）送入 ACCU 1 中
LN              //在 ACCU 1 中计算浮点数的自然对数，将结果存在 ACCU 1 中
AN      OV      //扫描状态字中的 OV 位，确定是否是“0”
JC      OK      //如果在执行此指令期间未出错，则跳转到 OK 跳转标签
BEU             //如果执行此指令时出错，则块无条件结束
OK：T   MD20    //将结果从 ACCU 1 传送到存储器双字 MD20
```

（10）SIN　计算浮点数（32 位）角度的正弦值

指令格式为：SIN

SIN（计算 32 位 IEEE -FP 浮点数角度的正弦）计算用弧度表示的角度的正弦。在累加器 1 中，角度必须以浮点数表示，结果存储在累加器 1 中。此指令影响 CC 1、CC 0、OV 和 OS 状态字位。累加器 2 的内容（以及具有 4 个累加器的 CPU 的累加器 3 和累加器 4 的内容）保持不变。

例：

```
L       MD10    //将存储器双字 MD10 的值（此值必须为浮点数格式）送到 ACCU 1 中
SIN             //在 ACCU 1 中计算浮点数的正弦值，将结果存储在 ACCU 1 中
T       MD20    //将结果从 ACCU 1 传送到存储器双字 MD20
```

（11）COS　计算浮点数（32 位）角度的余弦值

指令格式为：COS

COS（计算 32 位 IEEE -FP 浮点数角度的余弦）计算用弧度表示的角度的余弦值。在累加器 1 中，角度必须以浮点数表示，结果存储在累加器 1 中。此指令影响 CC 1、CC 0、OV 和 OS 状态字位。累加器 2 的内容（以及具有 4 个累加器的 CPU 的累加器 3 和累加器 4 的内容）保持不变。

例：

```
L      MD10   //将存储器双字 MD10 的值（此值必须为浮点数格式）送到 ACCU 1 中
COS           //在 ACCU 1 中计算浮点数的余弦值，将结果存储在 ACCU 1 中
T      MD20   //将结果从 ACCU 1 传送到存储器双字 MD20
```

（12）TAN　计算浮点数（32 位）角度的正切值

指令格式为：TAN

TAN（计算 32 位 IEEE -FP 浮点数角度的正切）计算用弧度表示的角度的正切值。在累加器 1 中，角度必须以浮点数表示，结果存储在累加器 1 中。此指令影响 CC 1、CC 0、OV 和 OS 状态字位。累加器 2 的内容（以及具有 4 个累加器的 CPU 的累加器 3 和累加器 4 的内容）保持不变。

例：

```
L      MD10   //将存储器双字 MD10 的值（此值必须为浮点数格式）送到 ACCU 1 中
TAN           //在 ACCU 1 中计算浮点数的正切值，将结果存储在 ACCU 1 中
AN     OV     //扫描状态字中的 OV 位，确定是否是“0”
JC     OK     //如果在执行 TAN 指令期间未出错，跳转到 OK 跳转标签
BEU           //如果执行 TAN 指令时出错，块无条件结束
OK：T  MD20   //将结果从 ACCU 1 传送到存储器双字 MD20
```

（13）ASIN　计算浮点数（32 位）的反正弦值

指令格式为：ASIN

ASIN（计算 32 位 IEEE -FP 浮点数的反正弦）在累加器 1 中计算浮点数的反正弦值，结果是以弧度表示的角度。此指令影响 CC 1、CC 0、OV 和 OS 状态字位。累加器 2 的内容（以及具有 4 个累加器的 CPU 的累加器 3 和累加器 4 的内容）保持不变。

例：

```
L      MD10   //将存储器双字 MD10 的值（此值必须为浮点数格式）送到 ACCU 1 中
ASIN          //在 ACCU 1 中计算浮点数的反正弦值，将结果存储在 ACCU 1 中
AN     OV     //扫描状态字中的 OV 位，确定是否是“0”
JC     OK     //如果在执行 ASIN 指令期间未出错，则跳转到 OK 跳转标签
BEU           //如果执行 ASIN 指令时出错，则块无条件结束
OK：T  MD20   //将结果从 ACCU 1 传送到存储器双字 MD20
```

（14）ACOS　计算浮点数（32 位）的反余弦值

指令格式为：ACOS

ACOS（计算 32 位 IEEE -FP 浮点数的反余弦值）在累加器 1 中计算浮点数的反余弦值，结果是用弧度表示的角度。此指令影响 CC 1、CC 0、OV 和 OS 状态字位。累加器 2 的内容（以及具有 4 个累加器的 CPU 的累加器 3 和累加器 4 的内容）保持不变。

例：

```
L      MD10   //将存储器双字 MD10 的值（此值必须为浮点数格式）送到 ACCU 1 中
ACOS          //在 ACCU 1 中计算浮点数的反余弦值，将结果存储在 ACCU 1 中
AN     OV     //扫描状态字中的 OV 位，确定是否是“0”
JC     OK     //如果在执行 ACOS 指令期间未出错，则跳转到 OK 跳转标签
```

BEU　　　　　　//如果执行 ACOS 指令时出错，则块无条件结束

OK：T　　MD20　　//将结果从 ACCU 1 传送到存储器双字 MD20

（15）ATAN　　计算浮点数（32 位）的反正切值

指令格式为：ATAN

ATAN（计算 32 位 IEEE -FP 浮点数的反正切值）在累加器 1 中计算浮点数的反正切值，结果是以弧度表示的角度。此指令影响 CC 1、CC 0、OV 和 OS 状态字位。累加器 2 的内容（以及具有 4 个累加器的 CPU 的累加器 3 和累加器 4 的内容）保持不变。

例：

```
L       MD10    //将存储器双字 MD10 的值（此值必须为浮点数格式）装载到 ACCU 1 中
ATAN            //在 ACCU 1 中计算浮点数的反正切值，结果存储在 ACCU 1 中
AN      OV      //扫描状态字中的 OV 位，确定是否是“0”
JC      OK      //如果在执行 ATAN 指令期间未出错，则跳转到 OK 跳转标签
BEU             //如果执行 ATAN 指令时出错，则块无条件结束
OK：T   MD20    //将结果从 ACCU 1 传送到存储器双字 MD20
```

2. 梯形图（LAD）的浮点运算指令

（1）ADD_ R　实数加

指令符号为：



EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN1（被加数）和 IN2（加数）的数据类型为 REAL 型，存储区为 I、Q、M、L、D 或常数。OUT（运算结果）的数据类型为 REAL 型，存储区为 I、Q、M、L、D。在起用（EN）输入端通过一个逻辑“1”来激活 ADD_ R（实数加）。IN1 和 IN2 相加，其结果通过 OUT 来查看。如果结果超出了浮点数允许的范围（溢出或下溢），OV 位和 OS 位将为“1”并且 ENO 为“0”，这样便不执行此数学框后由 ENO 连接的其他功能（层叠排列）。

例：

由 I0.0 处的逻辑“1”激活 ADD_ R 框。MD0 + MD4 相加的结果输出到 MD10。如果结果超出了浮点数的允许范围，或者如果没有处理该程序语句（I0.0 = 0），则设置输出 Q4.0。

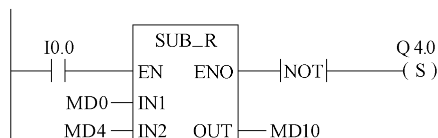
（2）SUB_ R　实数减

指令符号为：

EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN1（被减数）和 IN2（减数）的数据类型为 REAL 型，存储区为 I、Q、M、L、D 或常数。OUT（运算结果）的数据类型为 REAL 型，存储区为 I、Q、M、L、D。在起用（EN）输入端通过一个逻辑“1”来激活 SUB_ R（实数减）。从 IN1 中减去 IN2，并通过 OUT 查看结果。如果该结果超出了浮点数允许的范围（溢出或下溢），OV 位和 OS 位将为“1”并且 ENO 为逻辑“0”，这样便不执行此数学框后由 ENO 连接的其他函数（层叠排列）。

例：

在 10.0 处由逻辑“1”激活 SUB_ R 框。MD0 - MD4 相减的结果输出到 MD10。如果结果超出了浮点数



的允许范围，或者如果没有处理该程序语句，则设置输出 Q4.0。

（3）MUL_ R　实数乘

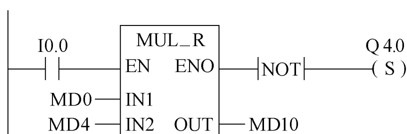
指令符号为：

MUL_R
EN　ENO
IN1
IN2　OUT

EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN1（被乘数）和 IN2（乘数）的数据类型为 REAL 型，存储区为 I、Q、M、L、D 或常数。OUT（运算结果）的数据类型为 REAL 型，存储区为 I、Q、M、L、D。在起用（EN）输入端通过一个逻辑“1”来激活 MUL_ R（实数乘）。IN1 和 IN2 相乘，结果通过 OUT 查看。如果该结果超出了浮点数允许的范围（溢出或下溢），OV 位和 OS 位将为“1”并且 ENO 为逻辑“0”，这样便不执行此数学框后由 ENO 连接的其他函数（层叠排列）。

例：

在 10.0 处由逻辑“1”激活 MUL_ R 框。MD0 × MD4 相乘的结果输出到 MD0。如果结果超出了浮点数的允许范围，或者如果没有处理该程序语句，则设置输出 Q4.0。



（4）DIV_ R　实数除

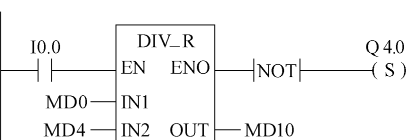
指令符号为：

DIV_R
EN　ENO
IN1
IN2　OUT

EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN1（被除数）和 IN2（除数）的数据类型为 REAL 型，存储区为 I、Q、M、L、D 或常数。OUT（运算结果）的数据类型为 REAL 型，存储区为 I、Q、M、L、D。在起用（EN）输入端通过一个逻辑“1”来激活 DIV_ R（实数除）。IN1 除以 IN2，其结果通过 OUT 查看。如果该结果超出了浮点数允许的范围（溢出或下溢），OV 位和 OS 位将为“1”并且 ENO 为逻辑“0”，这样便不执行此数学框后由 ENO 连接的其他函数（层叠排列）。

例：

由 I0.0 处的逻辑“1”激活 DIV_ R 框。MD0 除以 MD4 的结果输出到 MD10。如果结果超出了浮点数的允许范围，或者如果没有处理该程序语句，则设置输出 Q4.0。



（5）ABS　求浮点数的绝对值

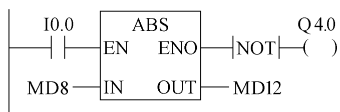
指令符号为：

ABS
EN　ENO
IN　OUT

EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN（输入）的数据类型为 REAL 型，存储区为 I、Q、M、L、D 或常数。OUT（运算结果）的数据类型为 REAL 型，存储区为 I、Q、M、L、D。ABS 求浮点数的绝对值。

例：

如果 10.0 = “1”，则 MD8 的绝对值在 MD12 输出。MD8 = +6.234 得到 MD12 = 6.234。如果未执行该转换（ENO = EN = 0），则输出 Q4.0 为“1”。

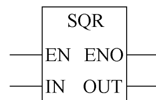


（6）SQR　求二次方

指令符号为：

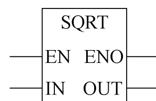
EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。

IN（输入）的数据类型为 REAL 型，存储区为 I、Q、M、L、D 或常数。OUT（运算结果浮点数的二次方）的数据类型为 REAL 型，存储区为 I、Q、M、L、D 或常数。SQR 求浮点数的二次方。



（7）SQRT　求平方根

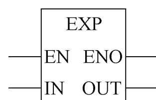
指令符号为：



EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN（输入）的数据类型为 REAL 型，存储区为 I、Q、M、L、D 或常数。OUT（运算结果浮点数的平方根）的数据类型为 REAL 型，存储区为 I、Q、M、L、D 或常数。SQRT 求浮点数的平方根。当地址大于“0”时，此指令得出一个正的结果。唯一例外的是：-0 的平方根是 -0。

（8）EXP　求指数值

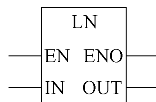
指令符号为：



EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN（输入）的数据类型为 REAL 型，存储区为 I、Q、M、L、D 或常数。OUT（运算结果浮点数的指数值）的数据类型为 REAL 型，存储区为 I、Q、M、L、D 或常数。EXP 求浮点数的以 e（=2，71828...）为底的指数值。

（9）LN　求自然对数

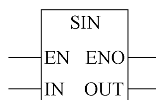
指令符号为：



EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN（输入）的数据类型为 REAL 型，存储区为 I、Q、M、L、D 或常数。OUT（运算结果浮点数的自然对数）的数据类型为 REAL 型，存储区为 I、Q、M、L、D 或常数。LN 求浮点数的自然对数。

（10）SIN　求正弦值

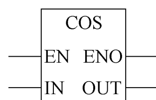
指令符号为：



EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN（输入）的数据类型为 REAL 型，存储区为 I、Q、M、L、D 或常数。OUT（运算结果浮点数的正弦）的数据类型为 REAL 型，存储区为 I、Q、M、L、D 或常数。SIN 求浮点数的正弦值。这里浮点数代表一个以弧度为单位的角度。

（11）COS　求余弦值

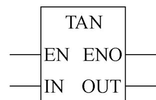
指令符号为：



EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN（输入）的数据类型为 REAL 型，存储区为 I、Q、M、L、D 或常数。OUT（运算结果浮点数的余弦）的数据类型为 REAL 型，存储区为 I、Q、M、L、D 或常数。COS 求浮点数的余弦值。这里浮点数代表一个以弧度为单位的角度。

（12）TAN　求正切值

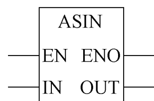
指令符号为：



EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN（输入）的数据类型为 REAL 型，存储区为 I、Q、M、L、D 或常数。OUT（运算结果浮点数的正切）的数据类型为 REAL 型，存储区为 I、Q、M、L、D 或常数。TAN 求浮点数的正切值。这里浮点数代表以弧度为单位的一个角度。

（13）ASIN　求反正弦值

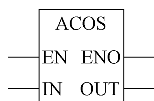
指令符号为：



EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN（输入）的数据类型为 REAL 型，存储区为 I、Q、M、L、D 或常数。OUT（运算结果浮点数的反正弦）的数据类型为 REAL 型，存储区为 I、Q、M、L、D 或常数。ASIN 求一个定义在 −1≤输入值≤1 范围内的浮点数的反正弦值。

（14）ACOS　求反余弦值

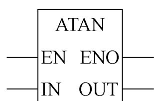
指令符号为：



EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN（输入）的数据类型为 REAL 型，存储区为 I、Q、M、L、D 或常数。OUT（运算结果浮点数的反余弦）的数据类型为 REAL 型，存储区为 I、Q、M、L、D 或常数。ACOS 求一个定义在 −1≤输入值≤1 范围内的浮点数的反余弦值。

（15）ATAN　求反正切值

指令符号为：



EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN（输入）的数据类型为 REAL 型，存储区为 I、Q、M、L、D 或常数。OUT（运算结果浮点数的反正切）的数据类型为 REAL 型，存储区为 I、Q、M、L、D 或常数。ATAN 求浮点数的反正切值。

九、装载和传送指令

可使用装载（L）和传送（T）指令进行编程，以在输入或输出模块与存储区之间、或在各存储区之间进行信息交换。CPU 在每个扫描周期中将这些指令作为无条件指令执行，也就是说，它们不受语句逻辑运算结果的影响。

1. 语句表（STL）的装载和传送指令

（1）L　　装载

指令格式为：　L <地址>

在将 ACCU 1 的原有内容保存到 ACCU 2 中，并将 ACCU 1 复位到“0”后，L <地址>会将被寻址的字节、字或双字装载到 ACCU 1 中。

例：

```
L    IB10      //将输入字节 IB10 装载到 ACCU 1 -L -L 中
L    MB120     //将存储器字节 MB120 装载到 ACCU 1 -L -L 中
L    DBB12     //将数据字节 DBB12 装载到 ACCU 1 -L -L 中
L    DIW15     //将实例数据字 DIW15 装载到 ACCU 1 -L 中
L    LD252     //将本地数据双字 LD252 装载到 ACCU 1 中
L    P# I 8.7  //将指针装载到 ACCU 1 中
L    OTTO      //将参数“OTTO”装载到 ACCU 1 中
```

（2）L STW　　将状态字装载到 ACCU 1 中

指令格式为：L STW

L STW（带有地址 STW 的指令 L）用状态字内容装载 ACCU 1。该指令的执行与状态位无关，对状态位也没有影响。注意对于 S7 -300 系列 CPU，L STW 语句不装载状态字的 FC、STA 和 OR 位。只有第 1、4、5、6、7 和第 8 位装载到累加器 1 低字的相应位中。

例：

```
L    STW       //将状态字内容装载到 ACCU 1 中
```

（3）LAR1　　从 ACCU 1 装载地址寄存器 1

指令格式为：LAR1

LAR1 用 ACCU 1 的内容（32 位指针）装载地址寄存器 AR1。ACCU 1 和 ACCU 2 保持不变。该指令的执行与状态位无关，对状态位也没有影响。

（4）LAR1 <D>　　用长整型（32 位指针）装载地址寄存器 1

指令格式为：LAR1 <D>

<D>的数据类型为 DWORD 或指针常量，存储区为 D、M、L。LAR1 <D> 用所寻址的双字 <D> 的内容或指针常量装载地址寄存器 AR1。ACCU 1 和 ACCU 2 保持不变。该指令的执行与状态位无关，对状态位也没有影响。

例 1：直接地址

LAR1　DBD20　　//用数据双字 DBD20 中的指针装载 AR1
LAR1　DID30　　//用实例数据双字 DID30 中的指针装载 AR1
LAR1　LD180　　//用本地数据双字 LD180 中的指针装载 AR1
LAR1　MD24　　//用存储器双字 MD24 的内容装载 AR1

例 2：指针常量

LAR1　P#M100.0　//用 32 位指针常量装载 AR1

（5）LAR1 AR2　　从地址寄存器 2 装载地址寄存器 1

指令格式为：LAR1 AR2

LAR1 AR2（带有地址 AR2 的指令 LAR1）用地址寄存器 AR2 的内容装载地址寄存器 1。ACCU 1 和 ACCU 2 保持不变。该指令的执行与状态位无关，对状态位也没有影响。

（6）LAR2　　从 ACCU 1 装载地址寄存器 2

指令格式为：LAR2

LAR2 用 ACCU 1 的内容（32 位指针）装载地址寄存器 AR2。ACCU 1 和 ACCU 2 保持不变。该指令的执行与状态位无关，对状态位也没有影响。

（7）LAR2 <D>　　用长整型（32 位指针）装载地址寄存器 2

指令格式为：LAR2 <D>

<D>的数据类型为 DWORD 或指针常量，存储区为 D、M、L。LAR1 <D> 用所寻址的双字 <D> 的内容或指针常量装载地址寄存器 AR2。ACCU 1 和 ACCU 2 保持不变。该指令的执行与状态位无关，对状态位也没有影响。

例 1：直接地址

LAR2　DBD 20　　//用数据双字 DBD20 中的指针装载 AR2
LAR2　DID 30　　//用实例数据双字 DID30 中的指针装载 AR2
LAR2　LD 180　　//用本地数据双字 LD180 中的指针装载 AR2
LAR2　MD 24　　//用存储器双字 MD24 中的指针装载 AR2

例 2：指针常量

LAR2　P#M100.0　//用 32 位指针常量装载 AR2

（8）T　　传送

指令格式为：T <地址>

T <地址>的数据类型为 BYTE、WORD、DWORD，寻址存储区为 I、Q、PQ、M、L、D。

如果主控继电器打开（MCR = 1），T <地址> 会将 ACCU 1 的内容传送（复制）到目标地址。如果 MCR = 0，则用 0 写目标地址。从 ACCU 1 复制的字节数取决于目标地址中表明的长

度。传送后，ACCU 1 还保存此数据。当传送至直接 I/O 区域（存储器类型 PQ）时，还将 ACCU 1 的内容或“0”（如果 MCR＝0）传送至过程映像输出表（存储器类型 Q）中的相应地址。该指令的执行与状态位无关，对状态位也没有影响。

例：

T　　QB10　　//将 ACCU 1 -L -L 的内容传送给输出字节 QB10

T　　MW14　　//将 ACCU 1 -L 的内容传送给存储器字 MW14

T　　DBD2　　//将 ACCU 1 的内容传送给数据双字 DBD2

（9）T STW　　将 ACCU 1 传送至状态字

指令格式为：T STW

T STW（带有地址 STW 的指令 T）将 ACCU 1 的 0～8 位传送给状态字。该指令的执行与状态位无关。

例：

T　　STW　　//将 ACCU 1 的 0～8 位传送给状态字

（10）CAR　　交换地址寄存器 1 和地址寄存器 2

指令格式为：CAR

CAR 交换地址寄存器 AR1 和 AR2 的内容。该指令的执行与状态位无关，对状态位也没有影响。地址寄存器 AR1 的内容移到地址寄存器 AR2 中，地址寄存器 AR2 的内容移到地址寄存器 AR1 中。

（11）TAR1　　将地址寄存器 1 传送至 ACCU 1

指令格式为：TAR1

TAR1 将地址寄存器 AR1 的内容传送给 ACCU 1（32 位指针）。ACCU 1 中的原有内容保存在 ACCU 2 中。该指令的执行与状态位无关，对状态位也没有影响。

（12）TAR1 <D>　　将地址寄存器 1 传送至目标地址（32 位指针）

指令格式为：TAR1 <D>

TAR1 <D> 将地址寄存器 AR1 的内容传送给寻址的双字 <D>。目标区域可以为存储器双字（MD）、本地数据双字（LD）、数据双字（DBD）和实例数据字（DID）。ACCU 1 和 ACCU 2 保持不变。该指令的执行与状态位无关，对状态位也没有影响。

例：

TAR1　　DBD20　　//将 AR1 的内容传送给数据双字 DBD20

TAR1　　DID30　　//将 AR1 的内容传送给实例数据双字 DID30

TAR1　　LD18　　//将 AR1 的内容传送给本地数据双字 LD18

TAR1　　MD24　　//将 AR1 的内容传送给存储器双字 MD24

（13）TAR1 AR2　　将地址寄存器 1 传送至地址寄存器 2

指令格式为：TAR1 AR2

TAR1 AR2（带有地址 AR2 的指令 TAR1）将地址寄存器 AR1 的内容传送给地址寄存器 AR2。ACCU 1 和 ACCU 2 保持不变。该指令的执行与状态位无关，对状态位也没有影响。

（14）TAR2　　将地址寄存器 2 传送至 ACCU 1

指令格式为：TAR2

TAR2 将地址寄存器 AR2 的内容传送给 ACCU 1（32 位指针）。ACCU 1 的内容提前保存到 ACCU 2 中。该指令的执行与状态位无关，对状态位也没有影响。

（15）TAR2 <D>　　将地址寄存器 2 传送至目标地址（32 位指针）

指令格式为：TAR2 ＜D＞

TAR2 ＜D＞ 将地址寄存器 AR2 的内容传送给寻址的双字 ＜D＞。目标区域可以为存储器双字（MD）、本地数据双字（LD）、数据双字（DBD）和实例双字（DID）。ACCU 1 和 ACCU 2 保持不变。该指令的执行与状态位无关，对状态位也没有影响。

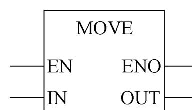
例：

```
TAR2    DBD20     //将 AR2 的内容传送给数据双字 DBD20
TAR2    DID30     //将 AR2 的内容传送给实例双字 DID30
TAR2    LD18      //将 AR2 的内容传送给本地数据双字 LD18
TAR2    MD24      //将 AR2 的内容传送给存储器双字 MD24
```

2. 梯形图（LAD）的装载和传送指令（MOVE 分配值）

（1）MOVE 分配值

指令符号为：



EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN（源值）的数据类型为所有长度为 8、16 或 32 位的基本数据类型，存储区为 I、Q、M、L、D 或常数。OUT（目标地址）的数据类型为所有长度为 8、16 或 32 位的基本数据类型，存储区为 I、Q、M、L、D。MOVE（分配值）通过起用 EN 输入来激活。在 IN 输入指定的值将复制到在 OUT 输出指定的地址。ENO 与 EN 的逻辑状态相同。MOVE 只能复制 BYTE、WORD 或 DWORD 数据对象。用户自定义数据类型（如数组或结构）必须使用系统功能“BLKMOVE”（SFC 20）来复制。

MCR（主站控制继电器）依存。只有当“传送”框位于激活的 MCR 区内时，才会激活 MCR 依存。在激活的 MCR 区内，如果开起了 MCR，同时有通往起用输入的电流，则按如上所述复制寻址的数据。如果 MCR 关闭，并执行了 MOVE，则无论当前 IN 状态如何，均会将逻辑“0”写入到指定的 OUT 地址。

注意：将某个值传送给不同长度的数据类型时，会根据需要截断或以零填充高位字节。

例 1：双字 1111 1111 0000 1111 1111 0000 0101 0101

传送　　结果

到双字：1111 1111 0000 1111 1111 0000 0101 0101

到字节：　　　　　　　　　　　0101 0101

到字：　　　　　　　　1111 0000 0101 0101

例 2：字节　　　　　　　　　　1111 0000

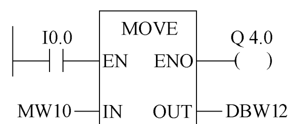
传送　　　结果

到字节：　　　　　　　　　　　1111 0000

到字：　　　　　　　　0000 0000 1111 0000

到双字：0000 0000 0000 0000 0000 0000 1111 0000

例：



如果 I0.0 为“1”，则执行该指令。把 MW10 的内容复制到当前打开 DB 的数据字 12。如果执行了该指令，则 Q4.0 为“1”。

如果该梯级在激活的 MCR 区之内：如果 MCR 开起，则按如上所述将 MW10 数据复制到 DBW12；如果 MCR 关闭，则将“0”写入到 DBW12。

（2）立即读取和立即写入

对于“立即读取”功能，必须按以下实例所示创建符号程序段。

对于对时间要求苛刻的应用程序，对数字输入的当前状态的读取可能要比正常情况下每 OB1 扫描周期一次的速度快。“立即读取”在扫描“立即读取”梯级时从输入模块中获取数字输入的状态。否则，必须等到下一 OB1 扫描周期结束，届时将以 P 存储器状态更新 I 存储区。

要从输入模块立即读取一个输入（或多个输入），请使用外设输入（PI）存储区来代替输入（I）存储区。可以字节、字或双字形式读取外设输入存储区。因此，不能通过触点（位）元素读取单一数字输入。

根据立即输入的状态有条件地传递电压。

1）CPU 读取包含相关输入数据的 PI 存储器的字。

2）如果输入位处于接通状态（为“1”），将对 PI 存储器的字与某个常数执行产生非零结果的 AND 运算。

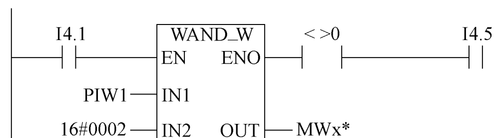
3）测试累加器的非零条件。

例：可以立即读取外设输入 I1.1 的梯形图程序段。

必须指定 * MWx，才能存储程序段。x 可以是允许的任何数。

WAND_ W 指令说明：

PIW1	0000000000101010
W#16#0002	0000000000000010
结果	0000000000000010



在此实例中，立即输入 I1.1 与 I4.1 和 I4.5 串联。

字 PIW1 包含 I1.1 的立即状态。对 PIW1 与 W#16#0002 执行 AND 运算。如果 PB1 中的 I1.1（第二位）为真（“1”），则结果不等于零。如果 WAND_ W 指令的结果不等于零，触点 A < >0 时将传递电压。

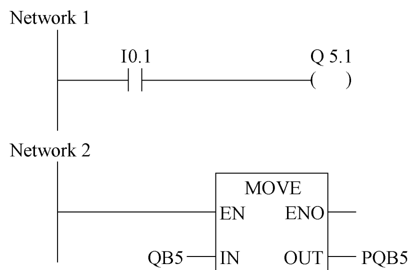
（3）立即写入

对于“立即写入”功能，必须按以下实例所示创建符号程序段。

对于对时间要求苛刻的应用程序，将数字输出的当前状态发送给输出模块的速度可能快于正常情况下在 OB1 扫描周期结束时发送一次的速度。“立即写入”将在扫描“立即写入”梯级时将数字输出写入输入模块。否则，必须等到下一 OB1 扫描周期结束，届时将以 P 存储器状态更新 Q 存储区。

要将一个输出（或多个输出）立即写入输出模块，请使用外设输出（PQ）存储区来代替输出（Q）存储区。可以字节、字或双字形式读取外设输出存储区。因此，不能通过线圈单元更新单一数字输出。要立即向输出模块写入数字输出的状态，将根据条件把包含相关位的 Q 存储器的字节、字或双字复制到相应的 PQ 存储器（直接输出模块地址）中。

注意：由于 Q 存储器的整个字节都写入了输出模块，因此在执行立即输出时，将更新该字节中的所有输出位。如果输出位在程序各处产生了多个中间状态（1/0），而这些状态不应发送给输出模块，则执行“立即写入”可能会导致危险情况（输出端产生瞬态脉冲）发生。作为常规设计原则，在程序中只能以线圈形式对外部输出模块引用一次。如果用户遵循此设计原则，则可以避免使用立即输出时的大多数潜在问题。



例：立即写入外设数字输出模块 5 通道 1 的等价梯形图程序段。

可以修改寻址输出 Q 字节（QB5）的状态位，也可以将其保持不变。程序段 1 中给 Q5.1 分

配 I0.1 信号状态。将 QB5 复制到相应的直接外设输出存储区（PQB5）。

字 PIW1 包含 I1.1 的立即状态。对 PIW1 与 W#16#0002 执行 AND 运算。如果 PB1 中的 I1.1（第二位）为真（“1”），则结果不等于零。如果 WAND_ W 指令的结果不等于零，触点 A < >0 时将传递电压。

在此实例中，Q5.1 为所需的立即输出位。

字节 PQB5 包含 Q5.1 位的立即输出状态。

MOVE（复制）指令还会更新 PQB5 的其他 7 位。

十、程序控制指令

程序控制指令是指块的调用和结束指令，用于执行程序控制。

1. 语句表（STL）的程序控制指令

（1）BE　　块结束

指令格式为：BE

BE（块结束）终止当前块中的程序扫描，并跳转到调用当前块的块中。在调用程序的块调用语句后的第一个指令处继续程序扫描。释放当前的本地数据区，上一个本地数据区开始成为当前本地数据区。重新打开调用块时打开的数据块。此外，恢复调用块的 MCR 相关性，并将 RLO 从当前块传送到调用当前块的块中。BE 与任何条件无关。然而，当跳过 BE 指令后，不结束当前程序扫描，而在块内的跳转目标处继续开始程序扫描。BE 指令与 S5 软件中的不完全相同。在 S7 硬件上使用时，本指令与 BEU 具有相同的功能。

例：

```
A       I1.0
JC      NEXT        //当 RLO =1 时（I 1.0 = 1），跳转到 NEXT 跳转标签
L       IW4         //当没有执行跳转时，在此继续执行
T       IW10
A       I6.0
A       I6.1
S       M12.0
BE                  //块结束
NEXT：NOP 0         //当执行了跳转时，在此继续执行
```

（2）BEC　　块有条件结束

指令格式为：BEC

如果 RLO = 1，那么 BEC（块有条件结束）中断当前块中的程序扫描，并跳转到调用当前块的块。在块调用后的第一个指令处继续程序扫描。释放当前的本地数据区，上一个本地数据区开始成为当前本地数据区。重新打开调用块时为当前数据块的数据块。恢复调用块的 MCR 相关性。将 RLO（= 1）从被终止的块传送到被调用的块。如果 RLO = 0，那么不执行 BEC。将 RLO 设置成 1，然后在 BEC 后的指令处继续程序扫描。

例：

```
A       I1.0        //更新 RLO
BEC                 //当 RLO = 1 时，结束块
L       IW4         //当 RLO = 0，不执行 BEC 时，在此继续执行
T       MW10
```

（3）BEU　　块无条件结束

指令格式为：BEU

BEU（块无条件结束）终止当前块中的程序扫描，并跳转到调用当前块的块。在块调用后的第一个指令处继续程序扫描。释放当前的本地数据区，上一个本地数据区开始成为当前本地数据区。重新打开调用块时打开的数据块。此外，恢复调用块的 MCR 相关性，并将 RLO 从当前块传送到调用当前块的块中。BEU 与任何条件无关。然而，当跳过 BEU 指令后，不结束当前程序扫描，而在块内的跳转目标处继续进行程序扫描。

例：

```
A I 1.0
JC      NEXT        //当 RLO =1 时（I 1.0 = 1），跳转到 NEXT 跳转标签
L       IW4         //当没有执行跳转时，在此继续执行
T       IW10
A       I6.0
A       I6.1
S       M12.0
BEU                 //块无条件结束
NEXT：NOP 0         //当执行了跳转时，在此继续执行
```

（4）CALL　　块调用

指令格式为：CALL <逻辑块标识符>

CALL <逻辑块标识符>用于调用功能（FC）或功能块（FB）、系统功能（SFC）或系统功能块（SFB）或调用由西门子提供的标准预编程块。CALL 指令调用作为地址输入的 FC 和 SFC 或 FB 或 SFB，与 RLO 或任何其他条件无关。当通过 CALL 调用 FB 或 SFB 时，必须给块提供一个相关的背景数据块。处理了被调用块后，调用块程序继续进行逻辑处理。可以指定逻辑块的绝对地址或符号地址。调用了 SFB/SFC 后，恢复寄存器的内容。

例：块调用语法见表 4 -8。

表 4-8　块调用语法

逻 辑 块	块 类 型	绝对地址调用语法
FC	功能	CALL FCn
SFC	系统功能	CALL SFCn
FB	功能块	CALL FBn1,DBn2
SFB	系统功能块	CALL SFBn1,DBn2

注意，使用 STL 编辑器时，表 4 -8 中的引用（n、n1 和 n2）必须指向有效的已存在块。同样，在使用前必须定义符号名。

传递参数（增量式编辑模式）。调用块可通过变量列表与被调用块交换参数。当输入一个有效的 CALL 语句时，自动在 STL 程序中扩展变量列表。当调用 FB、SFB、FC 或 SFC，而被调用块的变量声明表中具有 IN、OUT 和 IN_ OUT 声明时，这些变量以形式参数列表的形式添加到调用块中。

调用 FC 和 SFC 时，必须在调用逻辑块中将实际参数分配给形式参数。

调用 FB 和 SFB 时，只需指定通过上次调用改变的实际参数。处理了 FB 后，在背景数据块中存储实际参数。当实际参数是一个数据块时，必须指定完整的绝对地址，例如 DB1. DBW2。可以将 IN 参数指定为常数、绝对地址或符号地址。必须将 OUT 和 IN_ OUT 参数指定为绝对地址或符号地址。必须确保所有地址和常数都与要传送的数据类型兼容。CALL 将返回地址（选择器和

相对地址)、两个当前块的选择器以及 MA 位保存到 B（块）堆栈中。此外，CALL 取消激活 MCR 相关性，然后创建被调用块的本地数据区。

例 1：将参数分配给 FC6 调用

```
CALL   FC6
形式参数          实际参数
NO OF TOOL      ：=MW100
TIME OUT        ：= MW110
FOUND           ：= Q 0.1
ERROR           ：= Q 100.0
```

例 2：调用不带参数的 SFC

```
CALL     SFC43     //调用 SFC43，重新触发监视狗定时器（不带参数）
```

例 3：调用带背景数据块 DB1 的 FB99

```
CALL   FB99，DB1
形式参数     实际参数
MAX_ RPM        ：= #RPM1_ MAX
MIN_ RPM        ：= #RPM1
MAX_ POWER      ：= #POWER1
MAX_ TEMP       ：= #TEMP1
```

例 4：调用带背景数据块 DB2 的 FB99

```
CALL   FB99，DB2
形式参数          实际参数
MAX_ RPM        ：= #RPM2_ MAX
MIN_ RPM        ：= #RPM2
MAX_ POWER      ：= #POWER2
MAX_ TEMP       ：= #TEMP2
```

注意，每个 FB 或 SFC 调用必须具有一个背景数据块。在上面的例子中，必须在调用之前已经存在 DB1 和 DB2。

（5）调用 FB

指令格式为：CALL FB n1，DB n1

此指令用于调用用户自定义的功能块（FB)。CALL 指令调用作为地址输入的功能块，与 RLO 或其他条件无关。当使用 CALL 调用一个功能块时，必须给它提供一个背景数据块。处理了被调用块后，继续对调用块的程序进行处理。可以指定逻辑块的绝对地址或符号地址。

传递参数（增量式编辑模式）调用块可通过变量列表与被调用块交换参数。当输入一个有效的 CALL 指令时，自动在语句表程序中扩展变量列表。当调用一个功能块，而被调用块的变量声明表中具有 IN、OUT 和 IN_ OUT 声明时，这些变量以形式参数列表添加到调用块中。调用功能块时，只需指定必须通过上次调用改变的实际参数，因为在处理了功能块后，实际参数保存在背景数据块中。当实际参数是一个数据块时，必须指定完整的绝对地址，例如 DB1. DBW2。将 IN 参数指定为常数或作为绝对地址或符号地址。必须将 OUT 和 IN_ OUT 参数指定为绝对地址或符号地址。必须确保所有地址和常数都与要传送的数据类型兼容。CALL 将返回地址（选择器和相对地址)、两个当前块的选择器以及 MA 位保存到 B（块）堆栈中。此外，CALL 取消激活 MCR 相关性，然后创建被调用块的本地数据区。

例 1：带背景数据块 DB1 的 FB99 调用

CALL　FB99，DB1

形式参数　　　　实际参数

MAX_ RPM　　　：= #RPM1_ MAX

MIN_ RPM　　　：= #RPM1

MAX_ POWER　：= #POWER1

MAX_ TEMP　　：= #TEMP1

例 2：带背景数据块 DB2 的 FB99 调用

CALL　FB99，DB2

形式参数　　　　实际参数

MAX_ RPM　　　：= #RPM2_ MAX

MIN_ RPM　　　：= #RPM2

MAX_ POWER　：= #POWER2

MAX_ TEMP　　：= #TEMP2

注意，每个功能块的 CALL 必须具有一个背景数据块。在上面的实例中，必须在调用之前已经存在 DB1 和 DB2。

（6）调用 FC

指令格式为：CALL FC n

注意，当使用 STL 编辑器时，引用（n）必须与已存在的有效块相关。还必须在使用前定义符号名。该指令用于调用功能（FC）。CALL 指令调用作为地址输入的 FC，与 RLO 或其他条件无关。处理了被调用块后，继续对调用块的程序进行处理。可以指定逻辑块的绝对地址或符号地址。

传递参数（增量式编辑模式）调用块可通过变量列表与被调用块交换参数。当输入一个有效的 CALL 指令时，自动在语句表程序中扩展变量列表。当调用一个功能，而被调用块的变量声明表中具有 IN、OUT 和 IN_ OUT 声明时，这些变量以形式参数列表添加到调用块中。

调用功能时，必须在调用逻辑块中将实际参数分配给形式参数。可以将 IN 参数指定为常数、绝对地址或符号地址。必须将 OUT 和 IN_ OUT 参数指定为绝对地址或符号地址。必须确保所有地址和常数都与要传送的数据类型兼容。CALL 将返回地址（选择器和相对地址）、两个当前块的选择器以及 MA 位保存到 B（块）堆栈中。此外，CALL 取消激活 MCR 相关性，然后创建被调用块的本地数据区。

例：将参数分配给 FC6 调用

CALL　FC6

形式参数　　　　实际参数

NO OF TOOL　　：= MW100

TIME OUT　　　：= MW110

FOUND　　　　：= Q0.1

ERROR　　　　：= Q100.0

（7）调用 SFB

指令格式为：CALL SFB n1，DB n2

此指令用于调用由西门子提供的标准功能块（SFB）。CALL 指令调用作为地址输入的 SFB，与 RLO 或其他条件无关。当使用 CALL 调用一个系统功能块时，必须给它提供一个背景数据块。

处理了被调用块后，继续对调用块的程序进行处理。可以指定逻辑块的绝对地址或符号地址。

调用块可通过变量列表与被调用块交换参数。当输入一个有效的 CALL 指令时，自动在语句表程序中扩展变量列表。当调用一个系统功能块，而被调用块的变量声明表中具有 IN、OUT 和 IN_ OUT 声明时，这些变量以形式参数列表添加到调用块中。

调用系统功能块时，只需指定必须通过上次调用改变的实际参数，因为在处理了系统功能块后，实际参数保存在背景数据块中。当实际参数是一个数据块时，必须指定完整的绝对地址，例如 DB1. DBW2。

在进行参数传递时，可以将 IN 参数指定为常数、绝对地址或符号地址，此外必须将 OUT 和 IN_ OUT 参数指定为绝对地址或符号地址，必须确保所有地址和常数都与要传送的数据类型兼容。CALL 将返回地址（选择器和相对地址）、两个当前块的选择器以及 MA 位保存到 B（块）堆栈中。此外，CALL 取消激活 MCR 相关性，然后创建被调用块的本地数据区。

例：

```
CALL   SFB4，DB4
形式参数        实际参数
IN：            I0. 1
PT：            T#20s
Q：             M0. 0
ET：            MW10
```

注意，每个系统功能块 CALL 必须具有一个背景数据块。在上面的实例中，必须在调用之前已经存在 SFB4 和 DB4。

（8）调用 SFC

指令格式为：CALL SFC n

此指令用于调用由西门子提供的标准功能（SFC）。CALL 指令调用作为地址输入的 SFC，与 RLO 或其他条件无关。处理了被调用块后，继续对调用块的程序进行处理。可以指定逻辑块的绝对地址或符号地址。注意当使用 STL 编辑器时，引用（n）必须与已存在的有效块相关。还必须在使用前定义符号名。

传递参数（增量式编辑模式）

调用块可通过变量列表与被调用块交换参数。当输入一个有效的 CALL 指令时，自动在语句表程序中扩展变量列表。当调用一个系统功能，而被调用块的变量声明表中具有 IN、OUT 和 IN_ OUT 声明时，这些变量以形式参数列表添加到调用块中。调用系统功能时，必须在调用逻辑块中将实际参数分配给形式参数。在传递参数时，可以将 IN 参数指定为常数、绝对地址或符号地址。必须将 OUT 和 IN_ OUT 参数指定为绝对地址或符号地址，所有地址和常数都与要传送的数据类型兼容。

CALL 将返回地址（选择器和相对地址）、两个当前块的选择器以及 MA 位保存到 B（块）堆栈中。此外，CALL 取消激活 MCR 相关性，然后创建被调用块的本地数据区。

例：不带参数调用 SFC

```
CALL   SFC43      //调用 SFC43，重新触发监视狗定时器（不带参数）
```

（9）调用多重背景

指令格式为：CALL # 变量名

通过声明一个具有功能块数据类型的静态变量，创建一个多重背景。只在程序元素目录中包括已经声明的多重背景。

（10）调用库中的块

在 STEP 7 中可使用 SIMATIC 管理器中可供使用的库来选择集成在 CPU 操作系统（“标准库”）中的块和保存在库中供再次使用的块。

（11）CC　　条件调用

指令格式为：CC <逻辑块标识符>

CC <逻辑块标识符>（块条件调用）在 RLO＝1 时调用一个逻辑块。CC 用于不带参数调用 FC 或 FB 类型的逻辑块。除了不能使用调用程序传送参数之外，CC 的使用方法同 CALL 指令。该指令将返回地址（选择器和相对地址）、两个当前数据块的选择器以及 MA 位保存到 B（块）堆栈中，取消激活 MCR 相关性，创建被调用块的本地数据区，然后开始执行被调用代码。可以指定逻辑块的绝对地址或符号地址。

例：

A　　I2.0　　//检查输入 I 2.0 上的信号状态

CC　　FC6　　//当 I 2.0 为“1”时，调用功能 FC6

A　　M3.0　　//在被调用功能（I 2.0 ＝ 1）返回时执行，或当 I 2.0 ＝ 0 时，直接从 A I 2.0 语句后执行

注意，当 CALL 指令调用一个功能块（FB）或系统功能块（SFB）时，必须在语句中指定一个背景数据块（DB 号）。对于通过 CC 指令进行的一个调用，不能在语句中将数据块分配给地址。根据正在使用的程序段，程序编辑器在将梯形图编程语言转换成语句表编程语言期间，生成 UC 指令或 CC 指令。相反，应该尝试使用 CALL 指令，以避免在程序中发生错误。

（12）UC　　无条件调用

指令格式为：UC <逻辑块标识符>

UC <逻辑块标识符>（块无条件调用）调用 FC 或 SFC 类型的一个逻辑块。除了不能通过被调用块传送参数外，UC 如同 CALL 指令。该指令将返回地址（选择器和相对地址）、两个当前数据块的选择器以及 MA 位保存到 B（块）堆栈中，取消激活 MCR 相关性，创建被调用块的本地数据区，然后开始执行被调用代码。

例 1：

UC　　FC6　　//调用功能 FC6（不带参数）

例 2：

UC　　SFC43　　//调用系统功能 SFC43（不带参数）

注意，CALL 指令用于功能块（FB）或系统功能块（SFB）时，在指令中显式表示一个背景数据块（DB 号）。使用 UC 指令进行调用时，不能在 UC 地址中关联数据块。根据正在使用的程序段，程序编辑器在将梯形图编程语言转换成语句表编程语言期间，生成 UC 指令或 CC 指令。相反，应该尝试使用 CALL 指令，以避免在程序中发生错误。

（13）MCR（主控继电器）

注意，为了防止人员伤害或财产损失，禁止用 MCR 代替硬连线的机械主控继电器实现紧急停止功能。

主控继电器（MCR）是一个继电器梯形图主开关，用于激励和取消激励能量。由下列位逻辑触发的指令和传送指令取决于 MCR：＝ <位>、S <位>、R <位>、T <字节>、T <字>、T <双字>。

当 MCR 为 0 时，使用 T 指令将 0 写入到存储器字节、字和双字中。S 和 R 指令保持现有数值不变。指令＝（赋值指令）在已寻址的位中写入“0”。

指令取决于 MCR 以及对 MCR 信号状态的反应，MCR 的信号状态为 0（“关闭”）时，= <位>写入 0（模拟当断开电压时，进入静止状态的一个继电器）；S <位>、R <位>不写入（模拟当断开电压时，仍然位于其当前状态的一个继电器）；T <字节>、T <字>T <双字>写入 0（模拟当断开电压时，生成数值 0 的一个部件）。MCR 的信号状态为 1（“打开”）时，正常处理。

1）MCR（ -MCR 区域开始,）MCR -MCR 区域结束。

由 1 位宽、8 位深的堆栈控制 MCR。只要所有 8 个条目等于 1，就激励 MCR。MCR（指令将 RLO 位复制到 MCR 堆栈中）指令从堆栈中删除最后一个条目，并将空出的位置设置成 1。MCR（和）MCR 指令必须始终成对使用。出现故障时，即，当出现 8 个以上连续的 MCR（指令，或当 MCR 堆栈为空时尝试执行）MCR 指令，将触发 MCRF 错误消息。

2）MCRA -激活 MCR 区域，MCRD -取消激活 MCR 区域。

MCRA 和 MCRD 必须始终成对使用。程序中位于 MCRA 和 MCRD 之间的指令取决于 MCR 位的状态。在 MCRA -MCRD 序列外编程的指令不取决于 MCR 的位状态。

必须使用被调用块中的 MCRA 指令，在各个块中对功能（FC）和功能块（FB）的 MCR 相关性进行编程。

（14）使用 MCR 功能的注意事项

1）请注意使用 MCRA 激活主控继电器的块。

2）取消激活 MCR 时，所有赋值（T，=）都在 MCR（和）MCR 之间的程序段中写入数值 0。

3）当 MCR 指令前的 RLO = 0 时，取消激活 MCR。

4）危险：PLC 处于 STOP 状态或未定义的运行系统特征！编译器还对在 VAR_ TEMP 中定义的临时变量后的局部数据进行写访问，以计算地址。这表示下列命令序列将把 PLC 设置成 STOP，或导致未定义的运行系统特征：

① 形式参数访问

访问 STRUCT、UDT、ARRAY、STRING 类型的复杂 FC 参数的组件；访问来自具有多重背景能力的块（V2 版本的块）的 IN_ OUT 区域的、STRUCT、UDT、ARRAY、STRING 类型的复杂 FB 参数的组件；当地址高于 8180.0 时，访问具有多重背景能力（V2 版本的块）的功能块的参数；在具有多重背景能力（V2 版本的块）的功能块中访问类型为 BLOCK_ DB 的参数，打开 DB0。任何后继数据访问将 CPU 设置成 STOP。T 0、C 0、FC0 或 FB0 始终用于 TIMER、COUNTER、BLOCK_ FC 和 LOCK_ FB。

② 参数传递，通过调用可传递参数

LAD/FBD：梯形图或功能块中的 T 分支和中线输出以 RLO = 0 开始。

纠正方法是释放上述命令，使其与 MCR 不相关：使用所述语句或程序段前面的 MCRD 指令，取消激活主控继电器；使用所述语句或程序段后的 MCRA 指令，重新激活主控继电器。

（15）MCR（ 在 MCR 堆栈中保存 RLO，开始 MCR　和 ）MCR　　结束 MCR

指令格式为：MCR（

　　　　　　）MCR

MCR（（打开一个 MCR 区域）在 MCR 堆栈上保存 RLO，然后打开一个 MCR 区域。MCR 区域是指令 MCR（和相应的指令）MCR 之间的指令。指令 MCR（必须始终与指令）MCR 一起使用。

当 RLO =1 时，MCR“打开”。正常执行该 MCR 区域内与 MCR 有关的指令。

当 RLO =0 时，MCR“关闭”。

指令取决于 MCR 的位状态，当 MCR 的信号状态为 0（“关闭”）时，= <位>写入 0（模拟当断开电压时，进入静止状态的一个继电器）；S <位>、R <位>不写入（模拟当断开电压时，仍然位于其当前状态的一个继电器）；T <字节>、T <字>、T <双字>写入 0（模拟当断开电压时，生成数值 0 的一个部件）。MCR 的信号状态为 1（“打开”）时，正常处理。

MCR（ 和 ）MCR 指令可以嵌套。最大嵌套深度为 8 层指令。可能的堆栈条目的最大数目为 8 个。当堆栈满时，执行 MCR（将产生 MCR 堆栈故障（MCRF）。

）MCR（结束 MCR 区域）从 MCR 堆栈中删除一个条目，然后结束一个 MCR 区域。释放最后一个 MCR 堆栈位置，并将其设置成 1。指令 MCR（必须始终与指令）MCR 一起使用。当堆栈空时，执行）MCR 将产生 MCR 堆栈故障（MCRF）。

例：

```
MCRA        //激活 MCR 区域
A   I1.0
MCR（       //在 MCR 堆栈中保存 RLO，然后打开 MCR 区域。当 RLO =1（I 1.0 = “1”）
            //时，MCR = “打开”；当 RLO =0（I 1.0 = “0”）时，MCR = “关闭”
A   I4.0
 =   Q8.0   //如果 MCR = “关闭”，则将 Q 8.0 置位为“0”，与 I 4.0 无关
L   MW20
T   QW10    //如果 MCR = “关闭”，则将“0”传送到 QW10 中
）MCR       //结束 MCR 区域
MCRD        //取消激活 MCR 区域
A   I1.1
 =   Q8.1   //这些指令位于 MCR 区域外，不取决于 MCR 位
```

（16）MCRA（激活 MCR 区域）和 MCRD（取消激活 MCR 区域）

指令格式为：MCRA
MCRD

MCRA（主控继电器激活）激励其后指令的 MCR 相关性。指令 MCRA 必须始终与指令 MCRD（主控继电器取消激活）一起使用。程序中位于 MCRA 和 MCRD 之间的指令取决于 MCR 位的信号状态。

MCRD（主控继电器取消激活）取消激励其后指令的 MCR 相关性。指令 MCRA（主控继电器激活）必须始终与指令 MCRD（主控继电器取消激活）组合使用。程序中位于 MCRA 和 MCRD 之间的指令取决于 MCR 位的信号状态。执行该指令，与状态字的位无关，也不影响状态字的位。

2. 梯形图（LAD）的程序控制指令

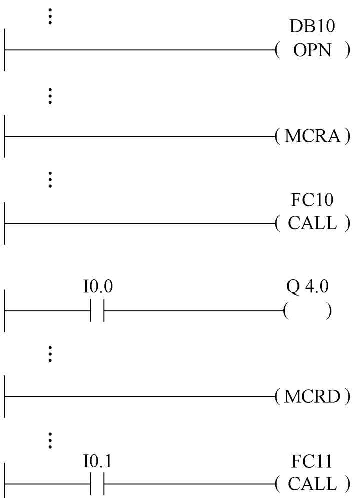
（1）---(Call)　调用来自线圈的 FC SFC（不带参数）

指令符号为：　<FC/SFC 编号>
---(CALL)

<FC/SFC 编号>的数据类型为 BLOCK_ FC 和 BLOCK_ SFC，FC/SFC 编号的范围取决于 CPU。---(Call)（不带参数调用 FC 或 SFC）用于调用没有传递参数的功能（FC）或系统功能（SFC）。只有在 CALL 线圈上 RLO 为“1”时，才执行调用。当执行 ---(Call)时，存储调用块的返回地址，由当前的本地数据区代替以前的本地数据区，将 MA 位（有效 MCR 位）移位到 B 堆栈中，为被调用的功能创建一个新的本地数据区。之后，在被调用的 FC 或 SFC 中继续进行程

序处理。

例：



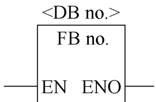
梯形图的梯级是由用户编写的功能块中的程序段。在该 FB 中，打开 DB10，并激活 MCR 功能。当执行无条件调用 FC10 时，发生下面情况：

保存调用 FB 的返回地址，并保存 DB10 和调用 FB 的背景数据块的选择数据。在 MCRA 指令中设置成“1”的 MA 位被推入到 B 堆栈中，然后为被调用块（FC10）将 MA 位设置成“0”。继续在 FC10 中进行程序处理。当 FC10 要求 MCR 功能时，必须在 FC10 内重新激活该功能。当完成 FC10 时，程序处理返回调用 FB。不管 FC10 使用了哪个 DB，都恢复 MA 位，DB10 和用户编写 FB 的背景数据块重新成为当前 DB。通过将 I0.0 的逻辑状态分配给输出 Q4.0，程序继续处理下一个梯级。FC11 的调用为条件调用。只有在 I0.1 为“1”时，才执行调用。执行调用后，将程序控制传递给 FC11 并从 FC11 返回程序控制的过程，与已经描述过的 FC10 的过程相同。

注意，返回调用块后，不是总会再次打开以前打开的 DB。

（2）CALL_ FB　调用来自框的 FB

指令符号为：



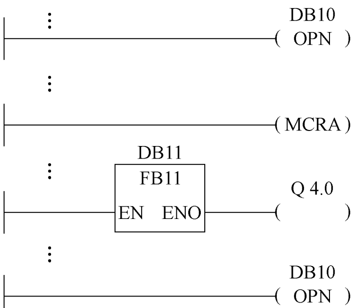
该符号取决于 FB（是否带参数以及带多少个参数）。它必须具有 EN、ENO 以及 FB 的名称或编号。

EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。EN（被减数）和 IN2（减数）的数据类型为 DINT 型，存储区为 I、Q、M、L、D 或常数。FB 编号的数据类型为 BLOCK_ FB，DB 号的数据类型为存储区为 BLOCK_ DB。

当 EN 为“1”时，执行 CALL_ FB（调用来自框的功能块）。当执行 CALL_ FB 时，存储调用块的返回地址，存储两个当前数据块（DB 和背景 DB）的选择数据，由当前的本地数据区代替以前的本地数据区，将 MA 位（有效 MCR 位）移位到 B 堆栈中，为被调用的功能块创建一个新的本地数据区。

之后，在被调用的功能块中继续进行程序处理。扫描 BR 位，以查找 ENO。用户必须使用---（SAVE）将所要求的状态（错误判断）分配给被调用块中的 BR 位。

例：



在该 FB 中，打开 DB10，并激活 MCR 功能。当执行无条件调用 FB11 时，发生下面情况：

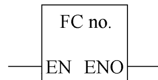
保存调用 FB 的返回地址，并保存 DB10 和调用 FB 的背景数据块的选择数据。在 MCRA 指令中设置成“1”的 MA 位被推入到 B 堆栈中，然后为被调用块（FB11）将 MA 位设置成“0”。继续在 FB11 中进行程序处理。当 FB11 要求 MCR 功能时，必须在 FB11 内重新激活该功能。必须使用指令 ---（SAVE）在 BR 位中保存 RLO 的状态，以便判断调用 FB 中的错误。当完成 FB11 时，程序处理返回调用 FB。恢复 MA 位，并重新打开用户编写的 FB 的背景数据块。当正确处理 FB11 时，ENO = “1”，因此 Q4.0 =

"1"。

注意，打开 FB 或 SFB 时，会丢失以前打开的 DB 的编号。必须重新打开所要求的 DB。

（3）CALL_ FC　调用来自框的 FC

指令符号为：



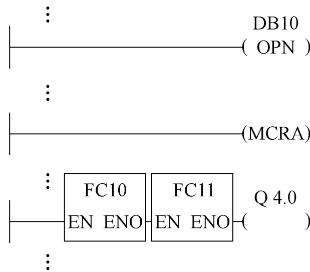
该符号取决于 FC（是否带参数以及带多少个参数）。它必须具有 EN、ENO 以及 FC 的名称或编号。

EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。FC 编号的数据类型为 BLOCK_ FC，CALL_ FC（调用来自框的功能）用于调用一个功能（FC）。当 EN 为"1"时，执行调用。如果执行了 CALL_ FC，存储调用块的返回地址，由当前的本地数据区代替以前的本地数据区，将 MA 位（有效 MCR 位）移位到 B 堆栈中，为被调用的功能创建一个新的本地数据区。之后，在被调用的功能中继续进行程序处理。

扫描 BR 位，以查找 ENO。必须使用 ---(SAVE）将所要求的状态（错误判断）分配给被调用块中的 BR 位。

当调用一个功能，而被调用块的变量声明表中具有 IN、OUT 和 IN_ OUT 声明时，这些变量以形式参数列表添加到调用块的程序中。当调用功能时，必须在调用位置处将实际参数分配给形式参数。功能声明中的任何初始值都没有含义。

例：

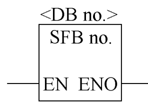


在 FB 中，打开 DB10，并激活 MCR 功能。当执行无条件调用 FC10 时，发生下面情况：

保存调用 FB 的返回地址，并保存 DB10 和调用 FB 的背景数据块的选择数据。在 MCRA 指令中设置成"1"的 MA 位被推入到 B 堆栈中，然后为被调用块（FC10）将 MA 位设置成"0"。继续在 FC10 中进行程序处理。当 FC10 要求 MCR 功能时，必须在 FC10 内重新激活该功能。必须使用指令 ---(SAVE）在 BR 位中保存 RLO 的状态，以便可以判断调用 FB 中的错误。当完成 FC10 时，程序处理返回调用 FB。恢复 MA 位。执行 FC10 后，根据 ENO，在调用 FB 中继续进行程序处理：ENO ="1" FC11 已处理，ENO = "0"在下一个程序段中开始处理。如果也正确处理了 FC11，则 ENO = "1"，因此 Q4.0 = "1"。注意，返回调用块后，不是总会再次打开以前打开的 DB。

（4）CALL_ SFB　调用来自框的系统 FB

指令符号为：

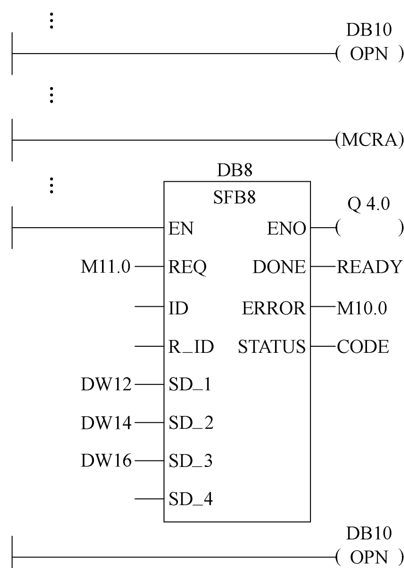


该符号取决于 SFB（是否带参数以及带多少个参数）。它必须具有 EN、ENO 以及 SFB 的名称或编号。EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。SFB 编号的数据类型为 BLOCK_ SFB，DB 号的数据类型为 BLOCK_ DB。-SFB 编号，范围取决于 CPU。

当 EN 为"1"时，执行 CALL_ SFB（调用来自框的系统功能块）。执行 CALL_ SFB 时，存储调用块的返回地址，存储两个当前数据块（DB 和背景 DB）的选择数据，由当前的本地数据区代替以前的本地数据区，将 MA 位（有效 MCR 位）移位到 B 堆栈中，为被调用的系统功能块创建一个新的本地数据区。然后在被调用的 SFB 中继续进行程序处理。当调用 SFB（EN = "1"）且没有发生错误时，ENO 为"1"。

例：

功能块中的程序段。在该 FB 中，打开 DB10，并激活 MCR 功能。当执行无条件调用 SFB8 时，发生下面情况：



保存调用 FB 的返回地址，并保存 DB10 和调用 FB 的背景数据块的选择数据。在 MCRA 指令中设置成“1”的 MA 位被推入到 B 堆栈中，然后为被调用块（SFB8）将 MA 位设置成“0”。继续在 SFB8 中进行程序处理。当完成 SFB8 时，程序处理返回调用 FB。恢复 MA 位，且用户编写的 FB 的背景数据块成为当前背景 DB。当正确处理 SFB8 时，ENO ＝ “1”，因此 Q4.0 ＝ “1”。

注意，打开 FB 或 SFB 时，会丢失以前打开的 DB 的编号。必须重新打开所要求的 DB。

（5）CALL_ SFC　调用来自框的系统 FC

指令符号为：

该符号取决于 SFC（是否带参数以及带多少个参数）。它必须具有 EN、ENO 以及 SFC 的名称或编号。

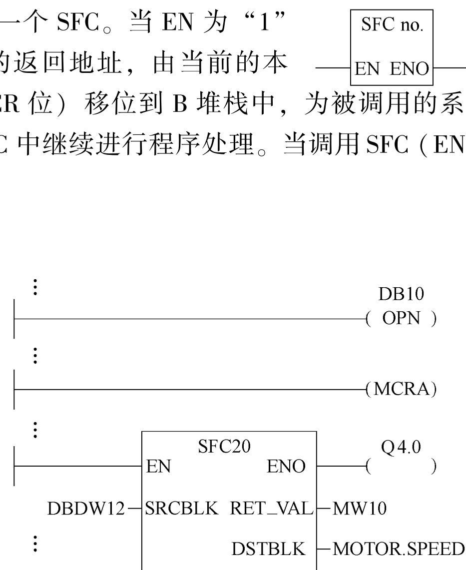
EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型。SFC 编号的数据类型为 BLOCK_ SFC。SFC 编号，范围取决于 CPU。

CALL_ SFC（调用来自框的系统功能）用于调用一个 SFC。当 EN 为“1”时，执行调用。当执行 CALL_ SFC 时，存储调用块的返回地址，由当前的本地数据区代替以前的本地数据区，将 MA 位（有效 MCR 位）移位到 B 堆栈中，为被调用的系统功能创建一个新的本地数据区。之后，在被调用的 SFC 中继续进行程序处理。当调用 SFC（EN = “1”）且没有发生错误时，ENO 为“1”。

SFC no.
EN ENO

例：

在该程序段的 FB 中，打开 DB10，并激活 MCR 功能。当执行无条件调用 SFC20 时，发生下面情况：



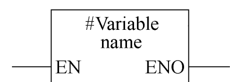
保存调用 FB 的返回地址，并保存 DB10 和调用 FB 的背景数据块的选择数据。在 MCRA 指令中设置成“1”的 MA 位被推入到 B 堆栈中，然后为被调用块（SFC20）将 MA 位设置成“0”。继续在 SFC20 中进行程序处理。当完成 SFC20 时，程序处理返回调用 FB。恢复 MA 位。处理 SFC20 后，根据 ENO，在调用 FB 中继续进行程序处理：当 ENO ＝ “1 时” Q4.0 = “1”，当 ENO = “0” 时，Q4.0 = “0”。

注意，返回调用块后，不是总会再次打开以前打开的 DB。

（6）调用多重背景

指令符号为：

EN（起用输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。# 变量名（多重背景的名称）的数据类型为 FB、SFB。



通过声明一个数据类型为功能块的静态变量，创建一个多重背景。只有已经声明的多重背景

才会包括在程序元素目录中。多重背景的符号改变取决于是否带参数以及带多少个参数。始终标出 EN、ENO 和变量名。

（7）调用来自库的块

在此，可使用 SIMATIC 管理器中可供使用的库来选择集成在 CPU 操作系统中的块和由于要多次使用而自行在库中保存的块。

（8）使用 MCR 功能的重要注意事项

1）注意：在 MCR 功能中使用 MCRA 激活主控制继电器的块，取消激活 MCR 时，在 ---(MCR <) 和 ---(MCR >) 之间的程序段中的所有赋值都写入数值 0。这对于包含一个赋值的所有框都有效，包括传递到块的参数在内。当 MCR < 指令之前的 RLO = 0 时，取消激活 MCR。

2）危险：PLC 处于 STOP 状态或未定义的运行特征时，编译器还对在 VAR_ TEMP 中定义的临时变量之后的局部数据进行写访问，以计算地址。这表示下列命令序列将把 PLC 设置成 STOP 状态，或导致未定义的运行特征：

① 形式参数访问

访问 STRUCT、UDT、ARRAY、STRING 类型的复杂 FC 参数的组件；访问来自具有多重背景能力的块（V2 版本的块）的 IN_ OUT 区域的 STRUCT、UDT、ARRAY、STRING 类型的复杂 FB 参数的组件；当地址高于 8180.0 时，访问具有多重背景能力（V2 版本的块）的功能块的参数；在具有多重背景能力（V2 版本的块）的功能块中访问类型为 BLOCK_ DB 的参数，打开 DB0，任何后继数据访问将 CPU 设置成 STOP 模式，T 0、C 0、FC0 或 FB0 始终用于 TIMER、COUNTER、BLOCK_ FC 和 LOCK_ FB。

② 参数传递，调用可传递参数的功能

LAD/FBD，梯形图或 FBD 中的 T 分支和中线输出以 RLO = 0 开始。

3）补救方法：释放上述命令，使其与 MCR 不相关。

① 在所述语句或程序段之前，使用主控制继电器取消激活指令，取消激活主控制继电器。

② 在所述语句或程序段之后，使用主控制继电器激活指令，重新激活主控制继电器。

（9）---(MCR <)　主控制继电器打开 和 -(MCR >)　主控制继电器关闭

主控制继电器打开的指令符号为：---(MCR <)

主控制继电器关闭的指令符号为：---(MCR >)

---(MCR <)（打开主控制继电器区域）在 MCR 堆栈中保存 RLO。---(MCR >)（关闭最后打开的 MCR 区域）从 MCR 堆栈中删除一个 RLO 条目。MCR 嵌套堆栈为 LIFO（后入先出）堆栈，且只能有 8 个堆栈条目（嵌套级别）。当堆栈已满时，---(MCR <) 功能产生一个 MCR 堆栈故障（MCRF）。下列元素与 MCR 有关，并在打开 MCR 区域时，受保存在 MCR 堆栈中的 RLO 状态的影响：

1）--(#)　中线输出

2）--( )　输出

3）--(S)　置位输出

4）--(R)　复位输出

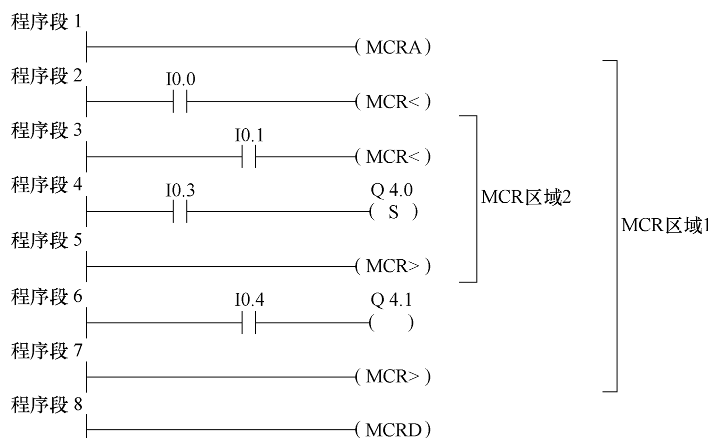
5）RS　复位触发器

6）SR　置位触发器

7）MOVE　赋值

例：

---(MCRA) 梯级激活 MCR 功能。然后可以创建至多 8 个嵌套 MCR 区域。在此实例中，有



两个 MCR 区域。第一个 ---(MCR >)(MCR 关闭）梯级属于第二个 ---(MCR <)(MCR 打开）梯级。两者之间的所有梯级都属于 MCR 区域 2。按如下方式执行这些功能：

I0.0 = “1”：将 I0.4 的逻辑状态分配给 Q4.1。

I0.0 = “0”：无论输入 I0.4 的逻辑状态如何，Q4.1 都为 0。

I0.1 = “1”：当 I0.3 为“1”时，将 Q4.0 设置成“1”。

I0.1 = “0”：无论 I0.3 的逻辑状态如何，Q4.0 都保持不变。

(10) ---(MCRA)　主控制继电器激活和 ---(MCRD)　主控制继电器取消激活

主控制继电器激活的指令符号为：---(MCRA)

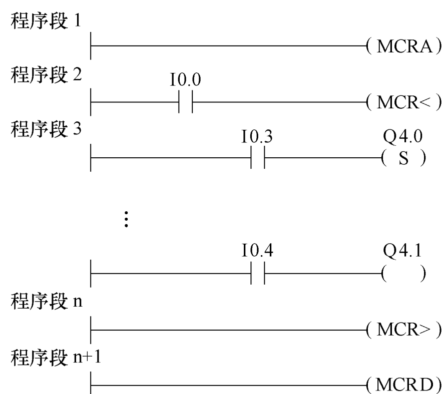
主控制继电器取消激活的指令符号为：---(MCRD)

---(MCRA)(激活主控制继电器）激活主控制继电器功能。在该命令后，可以使用 ---(MCR <) 和 ---(MCR >) 编程 MCR 区域。

---(MCRD)(取消激活主控制继电器）取消激活 MCR 功能。在该命令后，不能编程 MCR 区域。

例：

MCRA 梯级激活 MCR 功能。MCR < 和 MCR >(输出 Q4.0、Q4.1）之间的梯级按如下方式执行：



I0.0 = “1”（MCR 打开）：当 I0.3 为逻辑“1”时，将 Q4.0 设置成“1”，并将 I0.4 的逻辑状态分配给 Q4.1。

I0.0 = “0”（MCR 关闭）：无论 I0.3 的逻辑状态如何，Q4.0 保持不变，无论 I0.4 的逻辑状态如何，Q4.1 为“0”。

在下一个梯级中，指令 ---(MCRD) 取消激活 MCR。这表示不能再使用指令对 ---(MCR <) 和 ---(MCR >) 编程更多的 MCR 区域。

(11) ---(RET)　返回

指令符号为：---(RET)



RET（返回）用于有条件地退出块。对于该输出，要求在前面使用一个逻辑运算。

例：

当 I0.0 为“1”时，退出块。

十一、移位和循环移位指令

可使用移位指令逐位左移或右移累加器 1 中低字的内容或整个累加器的内容（参见 CPU 寄存器）。左移 n 位相当于将累加器的内容乘以“2^n”；右移 n 位相当于将累加器的内容除以“2^n”。例如，将以二进制格式表示的十进制数 3 左移 3 位时，在累加器中出现相当于十进制数 24 的二进制编码。将以二进制格式表示的十进制数 16 右移 2 位时，在累加器中出现相当于十进制数 4 的二进制编码。

移位指令后的数字或在累加器 2 的低字节中，低字节中的数值表示要移位的数目。由 0 或符号位的信号状态（0 代表正数、1 代表负数）填充移位指令空出的位。将最后一个移出的位装载到状态字的 CC 1 位中。复位状态字的 CC 0 和 OV 位为 0。可使用跳转指令来判断 CC 1 位。移位运算是无条件的，即它们的执行不需要任何特殊的条件，且不影响逻辑运算的结果。

可使用循环移位指令逐位左移或右移累加器 1 的整个内容（参见 CPU 寄存器）。循环移位指令触发与移位指令相似的功能。然而，它使用累加器中移出的位的信号状态填充空出的位。

循环移位指令后的数字或在累加器 2 的低字节中，低节中的数值表示要循环移位的数目。取决于指令的具体情况，循环移位也可以通过状态字的 CC 1 位进行。复位状态字的 CC 0 位为 0。

1. 语句表（STL）的移位和循环移位指令

（1）SSI　带符号整型移位（16 位）

指令格式为：SSI < 数目

SSI

< 数目 > 的数据类型为整型、无符号。要移位的位数目，范围为 0 ~ 15。

SSI < 数目 >：地址 < 数目 > 指定移位数目。允许的数值范围为 0 ~ 15。当 < 数目 > 大于 0 时，复位状态字的位 CC 0 和 OV 为 0。当 < 数目 > 等于 0 时，则将此移位指令视为 NOP 操作。

SSI：移位数目由 ACCU 2 -L -L 中的数值指定。可能的数值范围为 0 ~ 255。移位数目大于 16 时，始终产生相同的结果（ACCU 1 = 16#0000、CC 1 = 0 或 ACCU 1 = 16#FFFF、CC 1 = 1）。当移位数目大于 0 时，复位状态字的位 CC 0 和 OV 为 0。当移位数目为 0 时，则将移位指令视为 NOP 操作。

例：

内容	ACCU1 -H				ACCU1 -L			
位	31...	..	..	...16	15...	..	..	...0
执行 SSI 6 前	0101	1111	0110	0100	1001	1101	0011	1011
执行 SSI 6 后	0101	1111	0110	0100	1111	1110	0111	0100

例 1：

```
L      MW4      //将数值装载到 ACCU 1 中
SSI    6        //将 ACCU 1 中的带符号的位向右移动 6 位
T      MW8      //将结果传送到 MW8
```

例 2：

```
L      +3       //将数值 +3 装载到 ACCU 1 中
L      MW20     //将 ACCU 1 的内容送到 ACCU 2 中。将 MW20 的数值装载到 ACCU 1 中
SSI             //移位数目为 ACCU 2 -L -L 的数值≥将 ACCU 1 -L 中的带符号的位右
                //移 3 位；用符号位的状态填充空闲位
JP     NEXT     //当最后一个移出的位（CC 1）= 1 时，跳转到 NEXT 跳转标签
```

（2）SSD　　带符号长整型移位（32 位）

指令格式为：SSD

SSD ＜数目＞

＜数目＞的数据类型为整型、无符号。要移位的位数目，范围为 0～32。

SSD（右移带符号的长整型）逐位向右移动 ACCU 1 的整个内容。由符号位的信号状态填充移位指令空出的位。将最后一个移出的位装载到状态字的 CC 1 位中。地址＜数目＞或 ACCU 2 -L -L 中的数值指定要移位的位数目。

SSD ＜数目＞：地址＜数目＞指定移位数目。允许的数值范围为 0～32。当＜数目＞大于 0 时，复位状态字的位 CC 0 和 OV 为 0。当＜数目＞等于 0 时，则将移位指令视为 NOP 操作。

SSD：移位数目由 ACCU 2 -L -L 中的数值指定。可能的数值范围为 0～255。移位数目大于 32 时，始终产生相同的结果（ACCU 1 ＝ 32#00000000、CC 1 ＝ 0 或 ACCU 1 ＝ 32#FFFFFFFF、CC 1 ＝ 1）。当移位数目大于 0 时，复位状态字的位 CC 0 和 OV 为 0。当移位数目为 0 时，则将移位指令视为 NOP 操作。

例：

内容	ACCU1 -H				ACCU1 -L			
位	31...	..	..	...16	15...	..	..	...0
执行 SSD 7 前	1000	1111	0110	0100	0101	1101	0011	1011
执行 SSD 7 后	1111	1111	0001	1110	1100	1000	1011	1010

例 1：

```
L      MD4       //将数值装载到 ACCU 1 中
SSD    7         //根据符号，将 ACCU 1 中的位右移 7 位
T      MD8       //将结果传送到 MD8
```

例 2：

```
L      +3        //将数值 +3 装载到 ACCU 1 中
L      MD20      //将 ACCU 1 的内容装载到 ACCU 2 中。将 MD20 的数值送到 ACCU 1 中
SSD              //移位数目为 ACCU 2 -L -L 的数值≥将 ACCU 1 中的带符号的位右移
                 //3 位后，用符号位的状态填充空闲位
JP     NEXT      //当最后一个移出的位（CC 1）＝ 1 时，跳转到 NEXT 跳转标签
```

（3）SLW　　左移字（16 位）

指令格式为：SLW

SLW ＜数目＞

＜数目＞的数据类型为整型、无符号。要移位的位数目，范围为 0～15。

SLW（左移字）逐位向左移动 ACCU 1 -L 的内容。由零填充移位指令空出的位。将最后一个移出的位装载到状态字的 CC 1 位中。地址＜数目＞或 ACCU 2 -L -L 中的数值指定要移位的位数目。

SLW＜数目＞：地址＜数目＞指定移位数目。允许的数值范围为 0 ～15。当＜数目＞大于 0 时，复位状态字的位 CC 0 和 OV 为 0。当＜数目＞等于 0 时，则将此移位指令视为 NOP 操作。

SLW：移位数目由 ACCU 2 -L -L 中的数值指定。可能的数值范围为 0 ～ 255。移位数目大于 16 时，始终产生相同的结果：ACCU 1 -L ＝ 0、CC 1 ＝ 0、CC 0 ＝ 0 和 OV ＝ 0。当 0 ＜ 移位数目≤16 时，复位状态字的位 CC 0 和 OV 为 0。当移位数目为 0 时，则将移位指令视为 NOP 操作。

例：

```
内容              ACCU1 -H                        ACCU1 -L
位          31...  ..  ..  ...16              15...  ..  ..  ...0
执行 SLW 5 前  0101  1111  0110  0100     0101  1101  0011  1011
执行 SLW 5 后  0101  1111  0110  0100     1010  0111  0110  0000
```

例 1：

```
L       MW4       //将数值装载到 ACCU 1 中
SLW     5         //将 ACCU 1 中的位向左移动 5 位
T       MW8       //将结果传送到 MW8
```

例 2：

```
L       +5        //将数值 +5 装载到 ACCU 1 中
L       MW20      //将 ACCU 1 的内容装载到 ACCU 2 中。将 MW20 的数值送到 ACCU 1 中
SLW               //移位数目是 ACCU 2 -L -L 的数值≥将 ACCU 1 -L 中的位向左移动 5 位
JP      NEXT      //当最后一个移出的位（CC 1） =1 时，跳转到 NEXT 跳转标签
```

(4) SRW　右移字（16 位）

指令格式为：SRW

SRW <数目>

<数目>的数据类型为整型、无符号。要移位的位数目，范围为 0～15。

SRW（右移字）逐位向右移动 ACCU 1 -L 的内容。由零填充移位指令空出的位。将最后一个移出的位装载到状态字的 CC 1 位中。地址<数目>或 ACCU 2 -L -L 中的数值指定要移位的位数目。

SRW <数目>：地址<数目>指定移位数目。允许的数值范围为 0～15。当<数目>大于 0 时，复位状态字的位 CC 0 和 OV 为 0。当<数目>等于 0 时，则将此移位指令视为 NOP 操作。

SRW：移位数目由 ACCU 2 -L -L 中的数值指定。可能的数值范围为 0～255。移位数目大于 16 时，始终产生相同的结果：ACCU 1 -L = 0、CC 1 = 0、CC 0 = 0 和 OV = 0。当 0 < 移位数目≤16 时，复位状态字的位 CC 0 和 OV 为 0。当移位数目为 0 时，则将移位指令视为 NOP 操作。

例：

```
内容              ACCU1 -H                        ACCU1 -L
位          31...  ..  ..  ...16              15...  ..  ..  ...0
执行 SRW 6 前  0101  1111  0110  0100     0101  1101  0011  1011
执行 SRW 6 后  0101  1111  0110  0100     0000  0001  0111  0100
```

例 1：

```
L       MW4       //将数值装载到 ACCU 1 中
SRW     6         //将 ACCU 1 -L 中的位向右移动 6 位
T       MW8       //将结果传送到 MW8
```

例 2：

```
L       +3        //将数值 +3 装载到 ACCU 1 中
L       MW20      //将 ACCU 1 的内容装载到 ACCU 2 中。将 MW20 的数值送到 ACCU 1 中
SRW               //移位数目是 ACCU 2 -L -L 的数值≥将 ACCU 1 -L 中的位向右移动 3 位
SPP     NEXT      //当最后一个移出的位（CC 1）=1 时，跳转到 NEXT 跳转标签
```

(5) SLD　左移双字（32 位）

指令格式为：SLD

SLD <数目>

<数目>的数据类型为整型、无符号。要移位的位数目，范围为 0～32。

SLD（左移双字）逐位向左移动 ACCU 1 的整个内容。由零填充移位指令空出的位。将最后一个移出的位装载到状态字的 CC 1 位中。地址<数目>或 ACCU 2 -L -L 中的数值指定要移位的位数目。

SLD <数目>：地址<数目>指定移位数目。允许的数值范围为 0～32。当<数目>大于 0 时，复位状态字的位 CC 0 和 OV 为 0。当<数目>等于 0 时，则将此移位指令视为 NOP 操作。

SLD：移位数目由 ACCU 2 -L -L 中的数值指定。允许的数值范围为 0～255。移位数目大于 32 时，始终产生相同的结果：ACCU 1 = 0、CC 1 = 0、CC 0 = 0 和 OV = 0。当 0 < 移位数目≤32 时，复位状态字的位 CC 0 和 OV 为 0。当移位数目为 0 时，则将移位指令视为 NOP 操作。

例：

内容	ACCU1 -H				ACCU1 -L			
位	31...	..	..	...16	15...	..	..	...0
执行 SLD 5 前	0101	1111	0110	0100	0101	1101	0011	1011
执行 SLD 5 后	1110	1100	1000	1011	1010	0111	0110	0000

例 1：

```
L      MD4      //将数值装载到 ACCU 1 中
SLD    5        //将 ACCU 1 中的位向左移动 5 位
T      MD8      //将结果传送到 MD8
```

例 2：

```
L      +3       //将数值 +3 装载到 ACCU 1 中
L      MD20     //将 ACCU 1 的内容装载到 ACCU 2 中。将 MD20 的数值送到 ACCU 1 中
SLD             //移位数目是 ACCU 2 -L -L 的数值≥将 ACCU 1 中的位向左移动 3 位
JP     NEXT     //当最后一个移出的位（CC 1）=1 时，跳转到 NEXT 跳转标签
```

（6）SRD　右移双字（32 位）

指令格式为：SRD

SRD <数目>

<数目>的数据类型为整型、无符号。要移位的位数目，范围为 0～32。

SRD（右移双字）逐位向右移动 ACCU 1 的整个内容。由零填充移位指令空出的位。将最后一个移出的位装载到状态字的 CC 1 位中。地址<数目>或 ACCU 2 -L -L 中的数值指定要移位的位数目。

SRD <数目>：地址<数目>指定移位数目。允许的数值范围为 0～32。当<数目>大于 0 时，复位状态字的位 CC 0 和 OV 为 0。当<数目>等于 0 时，则将此移位指令视为 NOP 操作。

SRD：移位数目由 ACCU 2 -L -L 中的数值指定。允许的数值范围为 0～255。移位数目大于 32 时，始终产生相同的结果：ACCU 1 = 0、CC 1 = 0、CC 0 = 0 和 OV = 0。当 0 < 移位数目≤32 时，复位状态字的位 CC 0 和 OV 为 0。当移位数目为 0 时，则将移位指令视为 NOP 操作。

例：

内容	ACCU1 -H				ACCU1 -L			
位	31...	..	..	...16	15...	..	..	...0
执行 SRD 7 前	0101	1111	0110	0100	0101	1101	0011	1011
执行 SRD 7 后	0000	0000	1011	1110	1100	1000	1011	1010

例 1：

```
L       MD4      //将数值装载到 ACCU 1 中
SRD     7        //将 ACCU 1 中的位向右移动 7 位
T       MD8      //将结果传送到 MD8
```

例 2：

```
L       +3       //将数值 +3 装载到 ACCU 1 中
L       MD20     //将 ACCU 1 的内容装载到 ACCU 2 中。将 MD20 的数值送到 ACCU 1 中
SRD              //移位数目是 ACCU 2 -L -L 的数值≥将 ACCU 1 中的位向右移动 3 位
JP      NEXT     //当最后一个移出的位（CC 1）= 1 时，跳转到 NEXT 跳转标签
```

（7）RLD　循环左移双字（32 位）

指令格式为：RLD

RLD <数目>

<数目>的数据类型为整型、无符号。要循环移位的位数目，范围为 0～32。

RLD（循环左移双字）逐位向左循环移动 ACCU 1 的整个内容。循环移位指令空出的位由 ACCU 1 中移出位的信号状态填充。最后移出的位被装载到状态位 CC 1 中。而地址<数目>或 ACCU 2 -L -L 中的数值则指定要循环移位的位数目。

RLD <数目>：地址<数目>指定循环移位的数目。允许的数值范围为 0～32。当<数目>大于 0 时，将状态字的位 CC 0 和 OV 复位为 0。当<数目>等于 0 时，则将此循环移位指令视为 NOP 操作。

RLD：循环移位的数目由 ACCU 2 -L -L 中的数值指定。可能的数值范围为 0～255。当 ACCU 2 -L -L 内的数值大于 0 时，将状态字的位 CC 0 和 OV 复位为 0。如果循环移位的数目为零，则将此循环移位指令视为 NOP 操作。

例：

内容	ACCU1 -H				ACCU1 -L			
位	31...	..	..	...16	15...	..	..	...0
执行 RLD 4 前	0101	1111	0110	0100	0101	1101	0011	1011
执行 RLD 4 后	1111	0110	0100	0101	1101	0011	1011	0101

例 1：

```
L       MD2      //将数值装载到 ACCU 1 中
RLD     4        //将 ACCU 1 中的位向左循环移动 4 位
T       MD8      //将结果传送到 MD8
```

例 2：

```
L +4             //将数值 +3 装载到 ACCU 1 中
L       MD20     //将 ACCU 1 的内容装载到 ACCU 2 中。将 MD20 的数值送到 ACCU 1 中
RLD              //循环移位数目是 ACCU 2 -L -L 的值≥将 ACCU 1 中的位向左循环移动
                   4 位
JP      NEXT     //当最后一个移出的位（CC 1）= 1 时，跳转到 NEXT 跳转标签
```

（8）RRD　循环右移双字（32 位）

指令格式为：RRD

RRD <数目>

<数目>的数据类型为整型、无符号。要循环移位的位数目，范围为 0～32。

RRD（循环右移双字）逐位向右循环移动 ACCU 1 的整个内容。循环移位指令空出的位由 ACCU 1 中移出位的信号状态填充。最后移出的位被装载到状态位 CC 1 中。而地址 <数目> 或 ACCU 2 -L -L 中的数值则指定要循环移位的位数目。

RRD <数目>：地址 <数目> 指定循环移位的数目。允许的数值范围为 0 ~ 32。当 <数目> 大于 0 时，复位状态字的位 CC 0 和 OV 为 0。当 <数目> 等于 0 时，则将此循环移位指令视为 NOP 操作。

RRD：循环移位的数目由 ACCU 2 -L -L 中的数值指定。可能的数值范围为 0 ~ 255。当 ACCU 2 -L -L 的数值大于 0 时，复位状态字的位为 0。

例：

内容	ACCU1 -H				ACCU1 -L			
位	31...	..	..	...16	15...	..	..	...0
执行 RRD 4 前	0101	1111	0110	0100	0101	1101	0011	1011
执行 RRD 4 后	1011	0101	1111	0110	0100	0101	1101	0011

例 1：

```
L      MD2      //将数值装载到 ACCU 1 中
RRD    4        //将 ACCU 1 中的位向右循环移动 4 位
T      MD8      //将结果传送到 MD8
```

例 2：

```
L      +4       //将数值 +3 装载到 ACCU 1 中
L      MD20     //将 ACCU 1 的内容装载到 ACCU 2 中。将 MD20 的数值送到 ACCU 1 中
RRD             //循环移位数目是 ACCU 2 -L -L 的数值≥将 ACCU 1 中的位向右循环移
                  动 4 位
JP     NEXT     //当最后一个移出的位（CC 1）= 1 时，跳转到 NEXT 跳转标签
```

（9）RLDA　通过 CC 1 循环左移 ACCU 1（32 位）

指令格式为：RLDA

RLDA（通过 CC 1 循环左移双字）通过 CC 1 将 ACCU 1 的整个内容循环左移 1 位。复位状态字 CC 0 和 OV 为 0。

例：

内容		ACCU1 -H				ACCU1 -L			
位		31...	..	..	...16	15...	..	..	...0
执行 RLDA 前	×	0101	1111	0110	0100	0101	1101	0011	1011
执行 RLDA 后	0	1011	1110	1100	1000	1011	1010	0111	011X

（× =0 或 1，CC 1 之前的信号状态）

```
L      MD2      //将 MD2 的数值装载到 ACCU 1 中
RLDA            //通过 CC 1，将 ACCU 1 中的位循环左移 1 位
JP     NEXT     //当最后一个移出的位（CC 1）= 1 时，跳转到 NEXT 跳转标签
```

（10）RRDA　通过 CC 1 循环右移 ACCU 1（32 位）

指令格式为：RRDA

RRDA（通过 CC 1 循环右移双字）将 ACCU 1 的整个内容循环向右移动 1 位。复位状态字的位 CC 0 和 OV 为 0。

例：

内容　　ACCU1 -H　　ACCU1 -L

位	31...	..	..	...16	15...	..	..	...0
执行 RRDA 前×	0101	1111	0110	0100	0101	1101	0011	1011
执行 RRDA 后 1	X010	1111	1011	0010	0010	1110	1001	1101

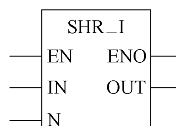
（× = 0 或 1，CC 1 的上一个信号状态）

```
L      MD2      //将 MD2 的数值装载到 ACCU 1 中
RRDA            //通过 CC 1，将 ACCU 1 中的位循环右移 1 位
JP     NEXT     //当最后一个移出的位（CC 1） = 1 时，跳转到 NEXT 跳转标签
```

2. 梯形图（LAD）的移位和循环移位指令

（1）SHR_ I　整数右移

指令符号为：



EN（使能输入）和 ENO（起用输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN（要移位的值）的数据类型为 INT，存储区为 I、Q、M、L、D。N（要移动的位数）的数据类型为 WORD，存储区为 I、Q、M、L、D。OUT（移位指令的结果）的数据类型为 INT 型，存储区为 I、Q、M、L、D。

SHR_ I（整数右移）指令通过使能（EN）输入位置上的逻辑“1”来激活。SHR_ I 指令用于将输入 IN 的 0～15 位逐位向右移动。16～31 位不受影响。输入 N 用于指定移位的位数。如果 N 大于 16，命令将按照 N 等于 16 的情况执行。自左移入的、用于填补空出位的位置将被赋予位 15 的逻辑状态（整数的符号位），如图 4 -10 所示。这意味着，当该整数为正时，这些位将被赋值“0”，而当该整数为负时，则被赋值为“1”。可在输出 OUT 位置扫描移位指令的结果。如果 N 不等于 0，则 SHR_ I 会将 CC 0 位和 OV 位设为“0”。ENO 与 EN 具有相同的信号状态。

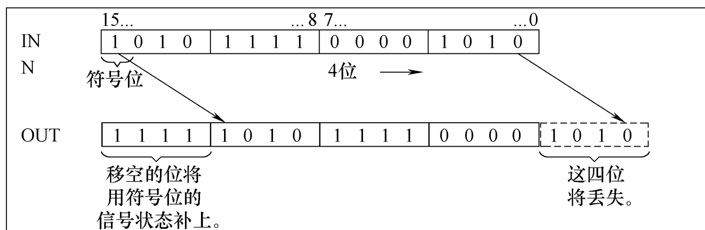
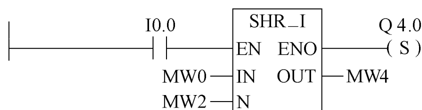


图 4 -10　整数右移

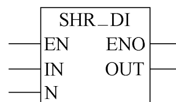
例：



SHR_ I 框由 I0. 0 位置上的逻辑“1”激活。装载 MW0 并将其右移由 MW2 指定的位数。结果将被写入 MW4。置位 Q4. 0。

（2）SHR_ DI　右移长整数

指令符号为：



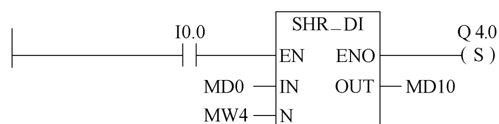
EN（使能输入）和 ENO（使能输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN（要移位的值）的数据类型为 DINT，存储区为 I、Q、M、L、D。N（要移动的位数）的数据类型为 WORD，存储区为 I、Q、M、L、D。OUT（移位指令的结果）的数据类型为 DINT 型，存储区为 I、Q、M、L、D。

SHR_ DI（右移长整数）指令通过使能（EN）输入位置上的逻辑“1”来激活。SHR_ DI 指令用于将输入 IN 的 0～31 位逐位向右移动。输入 N 用于指定移位的位数。如果 N 大于 32，命令

将按照 N 等于 32 的情况执行。自左移入的、用于填补空出位的位置将被赋予位 31 的逻辑状态（整数的符号位）。这意味着，当该整数为正时，这些位将被赋值“0”，而当该整数为负时，则被赋值为“1”。可在输出 OUT 位置扫描移位指令的结果。如果 N 不等于 0，则 SHR_ DI 会将 CC 0 位和 OV 位设为“0”。ENO 与 EN 具有相同的信号状态。

例

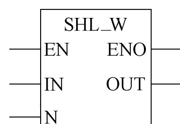
SHR_ DI 框由 I0.0 位置上的逻辑“1”激活。装载 MD0 并将其右移由 MW4 指定的位数。结果将被写入 MD10。置位 Q4.0。



（3）SHL_ W　字左移

指令符号为：

EN（使能输入）和 ENO（使能输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN（要移位的值）的数据类型为 WORD，存储区为 I、Q、M、L、D。N（要移动的位数）的数据类型为 WORD，存储区为 I、Q、M、L、D。OUT（字移位指令的结果）的数据类型为 WORD，存储区为 I、Q、M、L、D。



SHL_ W（字左移）指令通过使能（EN）输入位置上的逻辑“1”来激活。SHL_ W 指令用于将输入 IN 的 0 ~ 15 位逐位向左移动。16 ~ 31 位不受影响。输入 N 用于指定移位的位数。若 N 大于 16，此命令会在输出 OUT 位置上写入“0”，并将状态字中的 CC 0 和 OV 位设置为“0”。将自右移入 N 个零，用以补上空出的位置，如图 4 -11 所示。可在输出 OUT 位置扫描移位指令的结果。如果 N 不等于 0，则 SHL_ W 会将 CC 0 位和 OV 位设为“0”。ENO 与 EN 具有相同的信号状态。

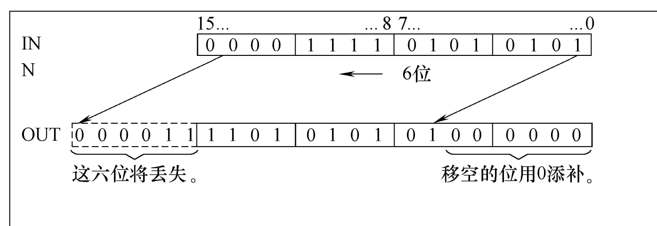
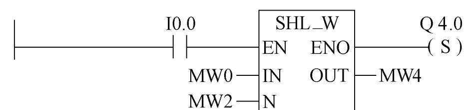


图 4 -11　整数左移

例：

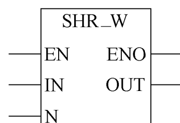
SHL_ W 框由 I0.0 位置上的逻辑“1”激活。装载 MW0 并将其左移由 MW2 指定的位数。结果将被写入 MW4。置位 Q4.0。



（4）SHR_ W　字右移

指令符号为：

EN（使能输入）和 ENO（使能输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN（要移位的值）的数据类型为 WORD，存储区为 I、Q、M、L、D。N（要移动的位数）的数据类型为 WORD，存储区为 I、Q、M、L、D。OUT（字移位指令的结果）的数据类型为 WORD，存储区为 I、Q、M、L、D。

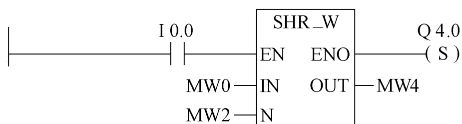


SHR_ W（字右移）指令通过使能（EN）输入位置上的逻辑“1”来激活。SHR_ W 指令用于将输入 IN 的 0 ~ 15 位逐位向右移动。16 ~ 31 位不受影响。输入 N 用于指定移位的位数。若 N 大于 16，此命令会在输出 OUT 位置上写入“0”，并将状态字中的 CC 0 位和 OV 位设置为“0”。

将自左移入 N 个零，用以补上空出的位置。可在输出 OUT 位置扫描移位指令的结果。如果 N 不等于 0，则 SHR_ W 会将 CC 0 位和 OV 位设为“0”。ENO 与 EN 有相同的信号状态。

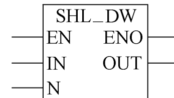
例：

SHR_ W 框由 I0. 0 位置上的逻辑“1”激活。装载 MW0 并将其右移由 MW2 指定的位数。结果将被写入 MW4。置位 Q4. 0。



（5）SHL_ DW 双字左移

指令符号为：

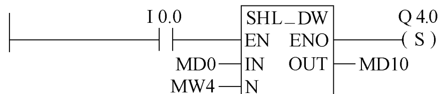


EN（使能输入）和 ENO（使能输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN（要移位的值）的数据类型为 DWORD，存储区为 I、Q、M、L、D。N（要移动的位数）的数据类型为 WORD，存储区为 I、Q、M、L、D。OUT（双字移位指令的结果）的数据类型为 DWORD，存储区为 I、Q、M、L、D。

SHL_ DW（双字左移）指令通过使能（EN）输入位置上的逻辑“1”来激活。SHL_ DW 指令用于将输入 IN 的 0 ~ 31 位逐位向左移动。输入 N 用于指定移位的位数。若 N 大于 32，此命令会在输出 OUT 位置上写入“0”并将状态字中的 CC 0 和 OV 位设置为“0”。将自右移入 N 个零，用以补上空出的位置。可在输出 OUT 位置扫描双字移位指令的结果。如果 N 不等于 0，则 SHL_ DW 会将 CC 0 位和 OV 位设为“0”。ENO 与 EN 具有相同的信号状态。

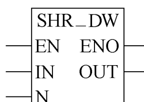
例：

SHL_ DW 框由 I0. 0 位置上的逻辑“1”激活。装载 MD0 并将其左移由 MW4 指定的位数。结果将被写入 MD10。置位 Q4. 0。



（6）SHR_ DW 双字右移

指令符号为：



EN（使能输入）和 ENO（使能输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN（要移位的值）的数据类型为 DWORD，存储区为 I、Q、M、L、D。N（要移动的位数）的数据类型为 WORD，存储区为 I、Q、M、L、D。OUT（双字移位指令的结果）的数据类型为 DWORD，存储区为 I、Q、M、L、D。

SHR_ DW（双字右移）指令通过使能（EN）输入位置上的逻辑“1”来激活。SHR_ DW 指令用于将输入 IN 的 0 ~ 31 位逐位向右移动。输入 N 用于指定移位的位数。若 N 大于 32，此命令会在输出 OUT 位置上写入“0”并将状态字中的 CC 0 位和 OV 位设置为“0”。将自左移入 N 个零，用以补上空出的位置，如图 4 -12 所示。可在输出 OUT 位置扫描双字移位指令的结果。如果 N 不等于 0，则 SHR_ DW 会将 CC 0 位和 OV 位设为“0”。ENO 与 EN 具有相同的信号状态。

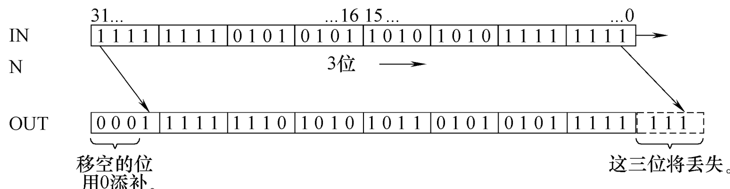
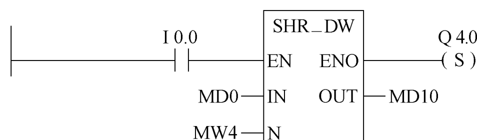


图 4 -12 双字右移

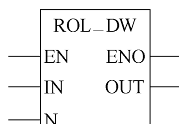
例：

SHR_ DW 框由 I0. 0 位置上的逻辑“1”激活。装载 MD0 并将其右移由 MW4 指定的位数。结果将被写入 MD10。置位 Q4. 0。



(7) ROL_DW 双字循环左移

指令符号为：



EN（使能输入）和 ENO（使能输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN（要循环移位的值）的数据类型为 DWORD，存储区为 I、Q、M、L、D。N（要循环移动的位数）的数据类型为 WORD，存储区为 I、Q、M、L、D。OUT（双字循环指令的结果）的数据类型为 DWORD，存储区为 I、Q、M、L、D。

ROL_DW（双字循环左移）指令通过使能（EN）输入位置上的逻辑“1”来激活。ROL_DW 指令用于将输入 IN 的全部内容逐位向左循环移位。输入 N 用于指定循环移位的位数。如果 N 大于 32，则双字 IN 将被循环移位（(N－1) 对 32 求模，所得的余数）+1 位。自右移入的位置将被赋予向左循环移出的各个位的逻辑状态，如图 4-13 所示。可在输出 OUT 位置扫描双字循环指令的结果。如果 N 不等于 0，则 ROL_DW 会将 CC 0 位和 OV 位设为“0”。ENO 与 EN 具有相同的信号状态。

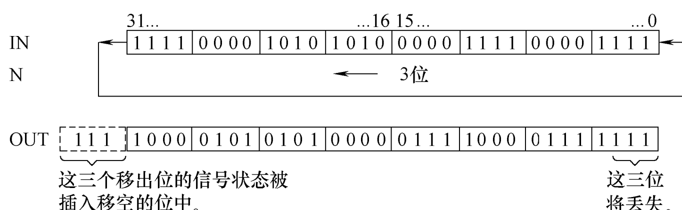
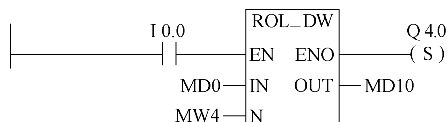


图 4-13 双字循环左移

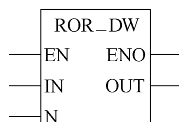
例：

ROL_DW 框由 I0.0 位置上的逻辑“1”激活。装载 MD0 并将其向左循环移位由 MW4 指定的位数。结果将被写入 MD10。置位 Q4.0。



(8) ROR_DW 双字循环右移

指令符号为：



EN（使能输入）和 ENO（使能输出）的数据类型为 BOOL 型，存储区为 I、Q、M、L、D。IN（要循环移位的值）的数据类型为 DWORD，存储区为 I、Q、M、L、D。N（要循环移动的位数）的数据类型为 WORD，存储区为 I、Q、M、L、D。OUT（双字循环指令的结果）的数据类型为 DWORD，存储区为 I、Q、M、L、D。

ROR_DW（双字循环右移）指令通过使能（EN）输入位置上的逻辑“1”来激活。ROR_DW 指令用于将输入 IN 的全部内容逐位向右循环移位。输入 N 用于指定循环移位的位数。如果 N 大于 32，则双字 IN 将被循环移位（(N－1）对 32 求模，所得的余数）+1 位。自左移入的位置将被赋予向右循环移出的各个位的逻辑状态，如图 4-14 所示。可在输出 OUT 位置扫描双字循环指令的结果。如果 N 不等于 0，则 ROR_DW 会将 CC 0 位和 OV 位置为“0”。ENO 与 EN 具有相同的信号状态。

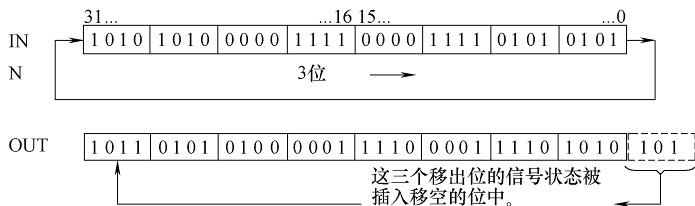
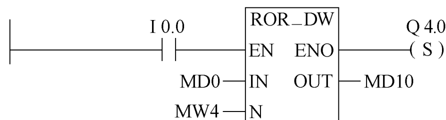


图 4-14 双字循环右移

例：

ROR_DW 框由 I0.0 位置上的逻辑“1”激活。装载 MD0 并将其向左循环移位由 MW4 指定的位数。结果将被写入 MD10。置位 Q4.0。



十二、状态位（LAD）指令

状态位指令属于位逻辑指令，用于对状态字的位进行处理。各状态位指令分别对下列条件之一做出反应，其中每个条件以状态字的一个或多个位来表示。

1）二进制结果位（BR---I I---）被置位（即信号状态为 1）。

2）数学运算函数发生溢出（OV---I I---）或存储溢出（OS---I I---）。

3）数学运算函数的结果是无序的（UO---I I---）。

4）数学运算函数的结果与 0 的关系有：= =0、< >0、>0、<0、> =0、< =0、起动和停止。

当状态位指令以串联方式连接时，该指令将根据“与”真值表将其信号状态校验的结果与前一逻辑运算结果合并。当状态位指令以并联方式连接时，该指令将根据“或”真值表将其结果与前一 RLO 合并。

状态字是 CPU 存储器中的一个寄存器，它包含可以在位逻辑指令和字逻辑指令的地址中引用的位。状态字的结构如图 4-15 所示。

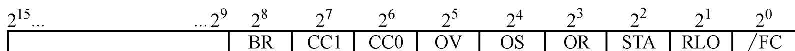
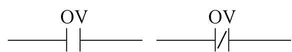


图 4-15 状态字的结构

可以通过整数数学运算函数、浮点数运算函数求状态字位的值。

（1）OV---I I --- 异常位溢出

指令符号为：



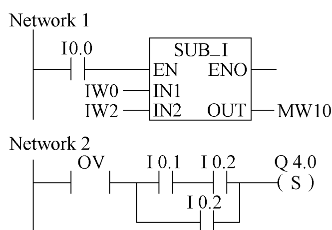
异常位溢出或异常位溢出取反的触点符号用于识别上次执行数学运算函数时的溢出。也就是

说，函数执行后指令的结果超出了允许的正、负范围。串联使用时，扫描的结果将通过 AND 与 RLO 链接；并联使用时，扫描结果通过 OR 与 RLO 链接。

例：

I0.0 的信号状态为“1”时将激活该框。如果数学运算函数“IW0 - IW2”的结果超出了允许的整数范围，则置位 OV 位。

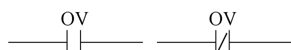
OV 的信号状态扫描为“1”。如果 OV 扫描的信号状态为“1”且程序段 2 的 RLO 为“1”，则置位 Q4.0。



注意，只有在有两个独立的程序段时，才需要 OV 扫描。否则，如果结果超出了允许的范围，则可以提取为“0”的数学运算函数的 ENO 输出。

（2）OS---|　|---　存储的异常位溢出

指令符号为：

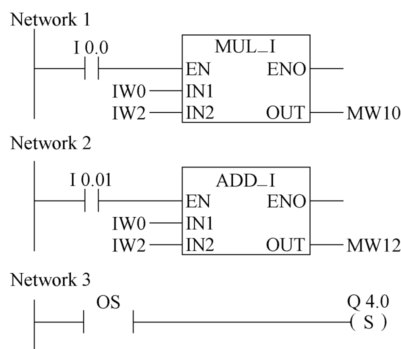


存储的异常位溢出或存储的异常位溢出取反的触点符号用于识别和存储数学运算函数中的锁存溢出。如果指令的结果超出了允许的负或正范围，则置位状态字中的 OS 位。与需要在执行后续数学运算函数前重写的 OV 位不同，OS 位在溢出发生时存储。OS 位将保持置位状态，直至离开该块。串联使用时，扫描的结果将通过 AND 与 RLO 链接；并联使用时，扫描结果通过 OR 与 RLO 链接。

例：

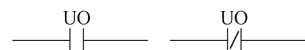
I0.0 的信号状态为“1”时将激活 MUL_I 框。I0.1 的逻辑为“1”时将激活 ADD_I 框。如果其中一个数学运算函数的结果超出了允许的整数范围，将把状态字中的 OS 位置位为“1”。如果 OS 扫描为逻辑“1”，则置位 Q4.0。

注意，只有在有两个独立的程序段时，才需要 OS 扫描。否则，可以提取第一个数学运算函数的 ENO 输出，并将其与第二个（层叠排列）数学运算函数的 EN 输入连接。



（3）UO---|　|---　无序异常位

指令符号为：

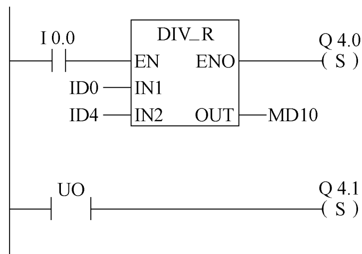


无序异常位或无序异常位取反的触点符号用于识别含浮点数的数学运算函数是否无序（也就是说，数学运算函数中的值是否有无效浮点数）。

如果含浮点数（UO）的数学运算函数的结果无效，则信号状态扫描为“1”。如果 CC 1 和 CC 0 中的逻辑运算显示“无效”，信号状态扫描的结果将是“0”。串联使用时，扫描的结果将通过 AND 与 RLO 链接；并联使用时，扫描结果通过 OR 与 RLO 链接。

例：

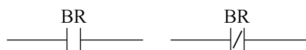
I0. 0 的信号状态为“1”时将激活该框。如果 ID0 或 ID4 的值为无效浮点数，则数学运算函数无效。如果 EN 的信号状态 = 1（激活）且在处理函数 DIV_R 时出错，则 ENO 的信号状态 = 0。



执行函数 DIV_R 时如果其中一个值不是有效的浮点数，将置位输出 Q4. 1。

（4）BR---|　　| ---　异常位二进制结果

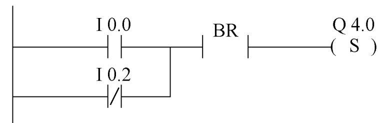
指令符号为：



异常位 BR 存储器或异常位 BR 存储器取反的触点符号用于测试状态字中 BR 位的逻辑状态。串联使用时，扫描结果将通过 AND 与 RLO 链接；并联使用时，扫描结果通过 OR 与 RLO 链接。BR 位用于字处理向位处理的转变。

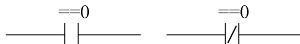
例：

如果 I0. 0 为“1”或 I0. 2 为“0”，且除此 RLO 外 BR 位的逻辑状态为“1”，则置位 Q4. 0。



（5） = = 0---|　　| ---　结果位等于 0

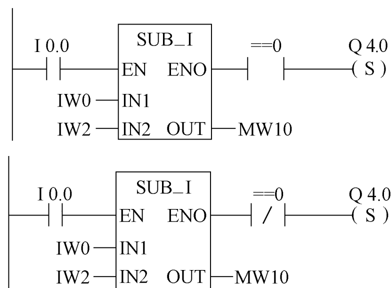
指令符号为：



结果位等于 0 或结果位取反后等于 0 的触点符号用于识别数学运算函数的结果是否等于“0”。指令扫描状态字的条件代码位 CC 1 和 CC 0，以确定结果与“0”的关系。串联使用时，扫描结果将通过 AND 与 RLO 链接；并联使用时，
扫描结果通过 OR 与 RLO 链接。

例：

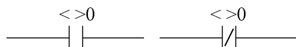
I0. 0 的信号状态为“1”时将激活该框。如果 IW0 的值等于 IW2 的值，数学运算函数“IW0 - IW2”的结果将等于“0”。如果函数得到正确执行且结果等于“0”，则置位 Q4. 0。



如果函数得到正确执行且结果不等于“0”，则置位 Q4. 0。

（6）< >0---|　　| ---　结果位不等于 0

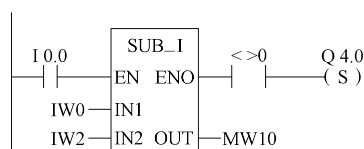
指令符号为：



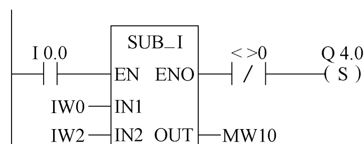
结果位不等于 0 或结果位取反后不等于 0 的触点符号用于识别数学运算函数的结果是否不等于“0”。指令扫描状态字的条件代码位 CC 1 和 CC 0，以确定结果与“0”的关系。串联使用时，扫描结果将通过 AND 与 RLO 链接；并联使用时，扫描结果通过 OR 与 RLO 链接。

例：

I0.0 的信号状态为“1”时将激活该框。如果 IW0 的值与 IW2 的值不同，数学运算函数“IW0-IW2”的结果将不等于“0”。如果函数得到正确执行且结果不等于“0”，则置位 Q4.0。

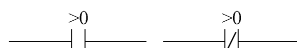


如果函数得到正确执行且结果等于“0”，则置位 Q4.0。



（7）>0---|　|---　结果位大于 0

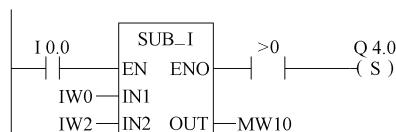
指令符号为：



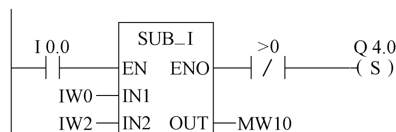
结果位大于 0 或结果位取反后大于 0 的触点符号用于识别数学运算函数的结果是否大于“0”。指令扫描状态字的条件代码位 CC 1 和 CC 0，以确定结果与“0”的关系。串联使用时，扫描结果将通过 AND 与 RLO 链接；并联使用时，扫描结果通过 OR 与 RLO 链接。

例：

I0.0 的信号状态为“1”时将激活该框。如果 IW0 的值大于 IW2 的值，数学运算函数“IW0-IW2”的结果将大于“0”。如果函数得到正确执行且结果大于“0”，则置位 Q4.0。

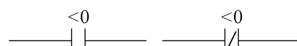


如果函数得到正确执行且结果不大于“0”，则置位 Q4.0。



（8）<0---|　|---　结果位小于 0

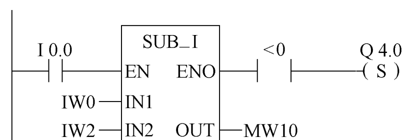
指令符号为：



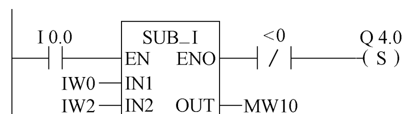
结果位小于 0 或结果位取反后小于 0 的触点符号用于识别数学运算函数的结果是否小于“0”。指令扫描状态字的条件代码位 CC 1 和 CC 0，以确定结果与“0”的关系。串联使用时，扫描结果将通过 AND 与 RLO 链接；并联使用时，扫描结果通过 OR 与 RLO 链接。

例：

I0.0 的信号状态为“1”时将激活该框。如果 IW0 的值小于 IW2 的值，数学运算函数“IW0-IW2”的结果将小于“0”。如果函数得到正确执行且结果小于“0”，则置位 Q4.0。

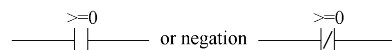


如果函数得到正确执行且结果不小于“0”，则置位 Q4.0。



（9）>=0---|　|---　结果位大于等于 0

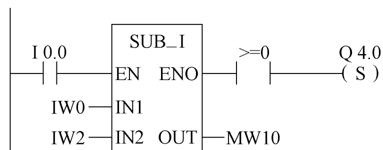
指令符号为：



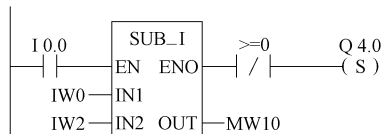
结果位大于等于0或结果位取反后大于等于0的触点符号用于识别数学运算函数的结果是否大于或等于“0”。指令扫描状态字的条件代码位 CC 1 和 CC 0，以确定结果与“0”的关系。串联使用时，扫描结果将通过 AND 与 RLO 链接；并联使用时，扫描结果通过 OR 与 RLO 链接。

例：

I0.0 的信号状态为“1”时将激活该框。如果 IW0 的值大于或等于 IW2 的值，数学运算函数“IW0-IW2”的结果将大于或等于“0”。如果函数得到正确执行且结果大于或等于“0”，则置位 Q4.0。

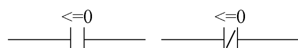


如果函数得到正确执行且结果不大于或等于“0”，则置位 Q4.0。



（10） < =0---|　|--- 结果位小于等于0

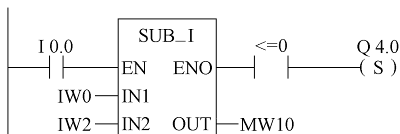
指令符号为：



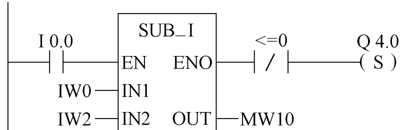
结果位小于等于0或结果位取反后小于等于0的触点符号用于识别数学运算函数的结果是否小于或等于“0”。指令扫描状态字的条件代码位 CC 1 和 CC 0，以确定结果与“0”的关系。串联使用时，扫描结果将通过 AND 与 RLO 链接；并联使用时，扫描结果通过 OR 与 RLO 链接。

例：

I0.0 的信号状态为“1”时将激活该框。如果 IW0 的值小于或等于 IW2 的值，数学运算函数“IW0-IW2”的结果将小于或等于“0”。如果函数得到正确执行且结果小于或等于“0”，则置位 Q4.0。



如果函数得到正确执行且结果不小于或等于“0”，则置位 Q4.0。



十三、定时器指令

S7 的 CPU 存储器中有一个为定时器保留的区域。此存储区域为每个定时器的地址保留一个16位字。

通过定时器指令和利用时钟定时更新定时器字可以访问定时器存储区域，在运行模式下，利用 CPU 中的时钟定时更新定时器字功能，可以按照由时间基准指定的间隔将给定的时间值递减一个单位，直到该时间值等于零为止。

定时器相当于继电器电路中的时间继电器，S7-300/400 的定时器分为脉冲定时器（SP）、扩展脉冲定时器（SE）、接通延时定时器（SD）、保持型接通延时定时器（SS）和断开延时定时器（SF）。

S_PULSE 为脉冲定时器，输出信号保持在1的最长时间与编程时间值 t 相同。如果输入信号变为0，则输出信号停留在1的时间会很短。

S_PEXT 为扩展脉冲定时器，输出信号在编程时间长度内始终保持在1，而与输入信号停留在1的时间长短无关。

S_ODT 为接通延时定时器，仅在编程时间到期，且输入信号仍为 1 时，输出信号变为 1。

S_ODTS 为带保持的接通延时定时器，输出信号仅在编程时间到期时才从 0 变为 1，而与输入信号停留在 1 的时间长短无关。

S_OFFDT 为断开延时定时器，在输入信号变为 1 或在定时器运行时，输出信号变为 1。当输入信号从 1 变为 0 时起动计时器。

定时器功能如图 4-16 所示。定时工作选择正确的定时器。

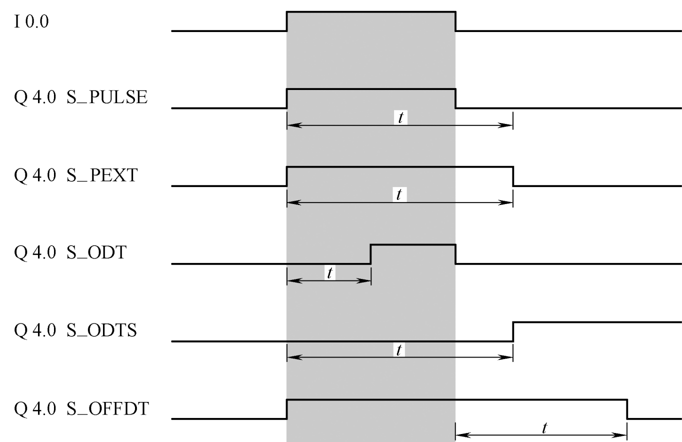


图 4-16　定时器功能

在 CPU 内部，时间值以二进制格式存放，占定时器字的 0 ~ 9 位。此时间值指定多个单位。时间更新可按照由时间基准指定的间隔将时间值递减一个单位。递减会持续进行，直至时间值等于零为止。可以在累加器 1 的低字中以二进制、十六进制或二进制编码的十进制（BCD）格式装入时间值。预装入时间值有以下两种格式。

第一种为 W#16#txyz，其中 t = 时间基准（即时间间隔或分辨率），此处 xyz = 以二进制编码的十进制格式表示的时间值。

第二种为 S5T#aH_bM_cS_dMS ，其中 H = 小时、M = 分钟、S = 秒、MS = 毫秒；用户变量为：a、b、c、d，时基是 CPU 自动选择的，选择的原则是在满足定时范围要求的条件下选择最小的时基。

可以输入的最大时间值是 9，990s 或 2H_46M_30S。

时间基准：定时器字的第 12 和 13 位包含二进制编码的时间基准。时间基准定义时间值以一个单位递减的间隔。最小的时间基准是 10ms，最大为 10s。时基代码为二进制数 00、01、10 和 11 时，对应的时基分别为 10ms、100ms、1s 和 10s。时基反映了定时器的分辨率，时基越小分辨率越高，可定时的时间越短。时基越大分辨率越低，可定时的时间越长。

定时器不接受超过 2 小时 46 分 30 秒的数值。对于范围限制（例如，2h10ms）而言，过高的分辨率将被截尾为有效分辨率。S5TIME 的通用格式对范围和分辨率有如下限制。

分辨率	范围
0. 01s	10MS 到 9S_990MS
0. 1s	100MS 到 1M_39S_900MS
1s	1S 到 16M_39S
10s	10S 到 2H_46M_30S

ACCU　　　1 中的位组态

当起动定时器时，ACCU1 的内容将被用作时间值。ACCU1-L 的 0 ~ 11 位保留二进制编码的十进制格式时间值（BCD 格式：由四位组成的每一组都包含一个十进制值的二进制代码）。第 12 和 13 位存放二进制编码的时间基准。

图 4-17 显示了装载定时器值 127 和 1s 时间基准的 ACCU1-L 的内容。

1. 语句表（STL）的定时器指令

（1）FR　　起用定时器（自由）

指令格式为：FR ＜定时器＞

＜定时器＞的数据类型为 TIMER，存储区为 T，定时器编号、范围取决于 CPU。

当 RLO 从 "0" 跳转到为 "1" 时，FR ＜定时器＞清除用于起动寻址定时器的边沿检测标记。起用指令（FR）前，RLO 位由 0 跳转到 1 即可起用定时器。

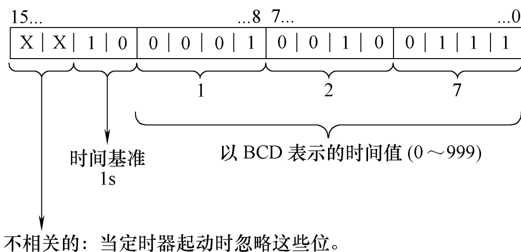


图 4-17　定时器值

无论是起动定时器还是正常的定时器指令，都不需要定时器的起用。起用只适用于重触发一个正在运行的定时器，即重新起动定时器。只有在 RLO = 1 的情况下继续处理起动指令时，才可进行重新起动。

例：

```
A    I2.0
FR   T1       //起用定时器 T1
A    I2.1
L  S5T#10s    //在 ACCU 1 中预置 10s
SP   T1       //起动定时器 T1 以作为脉冲定时器
A    I2.2
R  T1         //复位定时器 T1
A  T1         //检查定时器 T1 的信号状态
 =   Q4.0
L  T1         //以二进制的格式装载定时器 T1 的当前时间值
T  MW10
```

起用定时器时序图如图 4-18 所示。

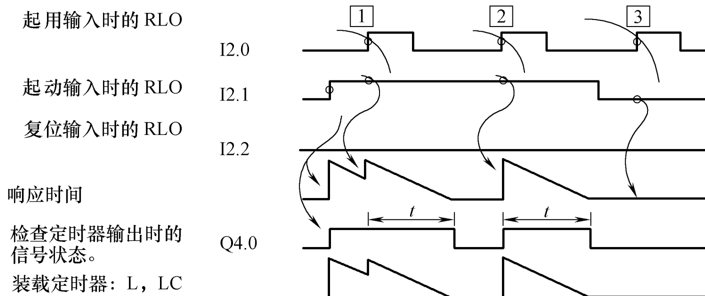


图 4-18　起用定时器时序图

1）在定时器运行的同时 RLO 在起用输入处从 0 变为 1，会完全重新起动定时器。程序时间将用作重新起动的当前时间。则 RLO 在起用输入处从 1 变为 0 将不会有任何作用。

2）如果在定时器未运行时 RLO 在起用输入处从 0 变为 1，且在使能输入处仍有一个值为 1 的 RLO，则定时器也会作为脉冲以已编程的时间起动。

3）当使能输入处仍有值为 1 的 RLO 时，则 RLO 在起用输入处从 0 变为 1 对定时器无影响。

（2）L　　将当前定时器值作为整数载入 ACCU 1

指令格式为：L ＜定时器＞

＜定时器＞的数据类型为 TIMER，存储区为 T，定时器编号、范围取决于 CPU。

在将 ACCU 1 的内容存入 ACCU 2 中后，L ＜定时器＞会从没有时间基准的寻址定时器字中以二进制整数的形式将当前定时器值装入 ACCU 1-L。

例：

L　T1　　//以二进制整数的形式为 ACCU 1 装载定时器 T1 的当前定时器值

将当前定时器值作为 ACCU1 如图 4-19 所示。

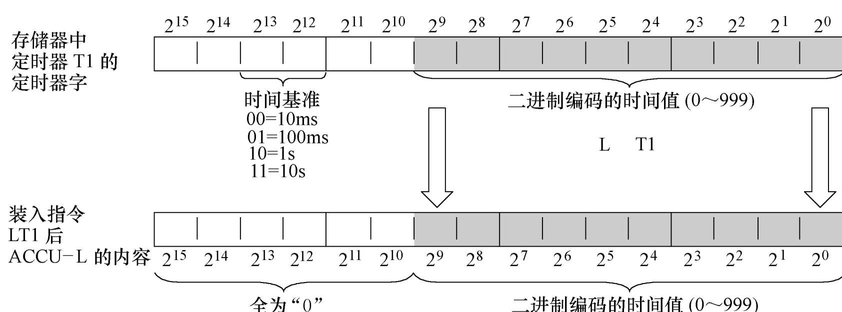


图 4-19　将当前定时器值作为 ACCU 1

注意，L ＜定时器＞ 只将当前定时器值的二进制代码装入 ACCU1-L，而不装载时间基准。装载的时间为初始值减去自定时器起动后所消耗的时间。

（3）LC　　将当前定时器值以 BCD 形式装入 ACCU 1

指令格式为：LC ＜定时器＞

＜定时器＞的数据类型为 TIMER，存储区为 T，定时器编号、范围取决于 CPU。在将 ACCU 1 的内容存入 ACCU 2 中后，LC ＜定时器＞会从寻址定时器字中以二进制编码的十进制（BCD）数的形式将当前定时器值和时间基准装入 ACCU 1 中。

例：

LC　T1　　//以 BCD 码的格式为 ACCU 1-L 装载定时器 T1 的时间基准和当前定时器值

当前定时器值以 BCD 形式装入 ACCU1 如图 4-20 所示。

（4）R　　复位定时器

指令格式为：R ＜定时器＞

＜定时器＞的数据类型为 TIMER，存储区为 T，定时器编号、范围取决于 CPU。

如果在 RLO 从“0”跳转到“1”，R＜定时器＞会停止当前计时功能并清除寻址定时器字的定时器值和时间基准。

例：

A　　I2. 1

R　　T1　　//检查输入 I 2. 1 的信号状态。如果 RLO 从 0 跳转到 1，则复位定时器 T1

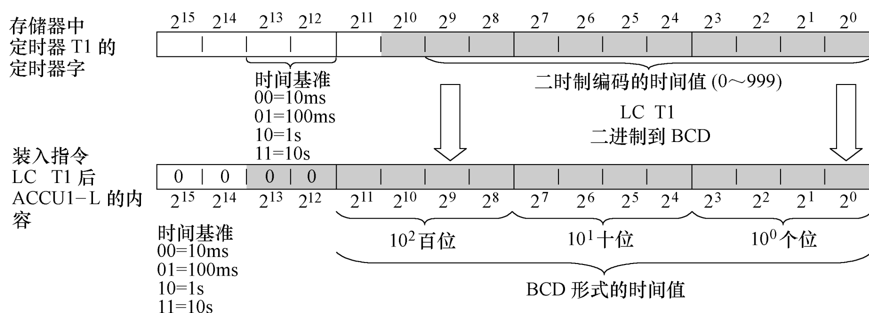


图 4-20 当前定时器值以 BCD 形式装入 ACCU 1

(5) SP 脉冲定时器

指令格式为：SP ＜定时器＞

＜定时器＞的数据类型为 TIMER，存储区为 T，定时器编号、范围取决于 CPU。SP ＜定时器＞ 在 RLO 从“0”跳转到“1”时起动寻址的定时器。只要 RLO＝1，程序时间间隔就会过去。如果在程序时间间隔截止之前 RLO 跳转到“0”，则停止计时器。此定时器起动指令要求将时间值和时间基准作为 BCD 数存储在 ACCU 1-L 中。

例：

```
A    I2.0
FR   T1        //起用定时器 T1
A    I2.1
L  S5T#10s     //在 ACCU 1 中预置 10s
SP   T1        //将定时器 T1 起动为脉冲定时器
A    I2.2
R    T1        //复位定时器 T1
A    T1        //检查定时器 T1 的信号状态
 =  Q4.0
L    T1        //以二进制形式装载定时器 T1 的当前时间值
T    MW10
LC   T1        //以 BCD 形式装载定时器 T1 的当前时间值
T  MW12
```

脉冲定时器时序图如图 4-21 所示。

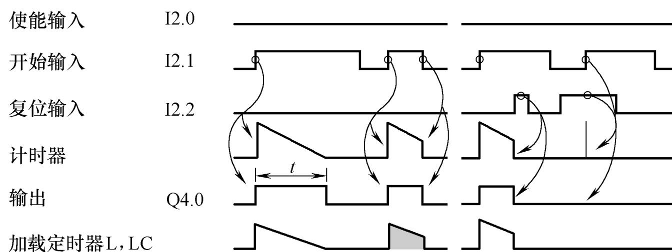


图 4-21 脉冲定时器时序图

(6) SE 扩展脉冲定时器

指令格式为：SE ＜定时器＞

<定时器>的数据类型为 TIMER，存储区为 T，定时器编号、范围取决于 CPU。SE <定时器> 在 RLO 从“0”跳转到“1”时起动寻址的定时器。程序时间间隔会流逝，即使 RLO 在这段时间内跳转到“0”。如果在程序时间间隔截止之前 RLO 从“0”跳转到“1”，则重新开始程序时间间隔。此定时器起动指令要求将时间值和时间基准作为 BCD 数存储在 ACCU 1-L 中。

例：

```
A     I2.0
FR    T1          //起用定时器 T1
A     I2.1
L   S5T#10s       //在 ACCU 1 中预置 10s
SE    T1          //将定时器 T1 起动为扩展脉冲定时器
A     I2.2
R     T1          //复位定时器 T1
A     T1          //检查定时器 T1 的信号状态
 =   Q4.0
L     T1          //以二进制形式装载定时器 T1 的当前定时器值
T     MW10
LC    T1          //以 BCD 形式装载定时器 T1 的当前定时器值
T   MW12
```

扩展的脉冲定时器时序图如图 4-22 所示。

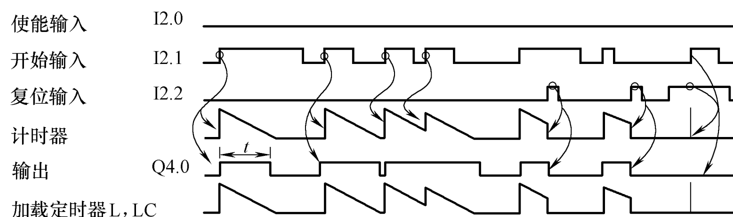


图 4-22　扩展的脉冲定时器时序图

（7）SD　　接通延时定时器

指令格式为：SD <定时器>

<定时器>的数据类型为 TIMER，存储区为 T，定时器编号、范围取决于 CPU。在 RLO 从“0”跳转到“1”时，SD <定时器>起动寻址的定时器。只要 RLO = 1，程序时间间隔就会流逝。如果在程序时间间隔截止之前 RLO 跳转到“0”，则停止计时。此定时器起动指令要求将时间值和时间基准作为 BCD 数存储在 ACCU 1-L 中。

例：

```
A     I2.0
FR    T1          //起动定时器 T1
A     I2.1
L   S5T#10s       //在 ACCU 1 中预置 10s
SD    T1          //起动定时器 T1 作为接通延时定时器
A     I2.2
R     T1          //复位定时器 T1
```

```
A     T1        //检查定时器 T1 的信号状态
 =   Q4. 0
L     T1        //以二进制形式装载定时器 T1 的当前定时器值
T     MW10
LC    T1        //以 BCD 形式装载定时器 T1 的当前定时器值
T   MW12
```

接通延时定时器时序图如图 4-23 所示。

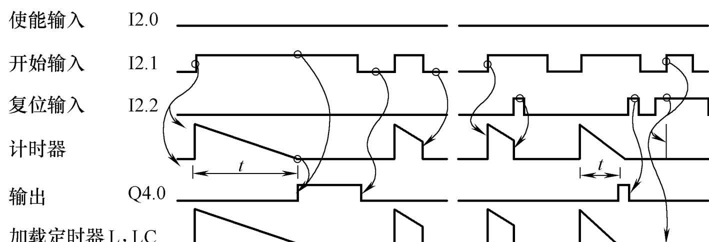


图 4-23　接通延时定时器时序图

（8）SS　　带保持的接通延时定时器

指令格式为：SS <定时器>

<定时器>的数据类型为 TIMER，存储区为 T，定时器编号、范围取决于 CPU。SS <定时器>（将定时器起动为带保持的接通定时器）在 RLO 从“0”跳转到“1”时起动寻址的定时器。完整的程序时间间隔会流逝，即使 RLO 在这段时间内跳转到“0”。如果在程序时间间隔截止之前 RLO 从“0”跳转到“1”，则重新触发程序时间间隔（重新起动）。此定时器指令要求将时间值和时间基准作为 BCD 数存储在 ACCU 1-L 中。

例：

```
A     I2. 0
FR    T1        //起用定时器 T1
A     I2. 1
L   S5T#10s     //在 ACCU 1 中预置 10s
SS    T1        //将定时器 T1 起动为带保持的接通延时定时器
A     I2. 2
R     T1        //复位定时器 T1
A     T1        //检查定时器 T1 的信号状态
 =   Q 4. 0
L     T1        //以二进制形式装载定时器 T1 的当前时间值
T     MW10
LC    T1        //以 BCD 形式装载定时器 T1 的当前时间值
T     MW12
```

带保持的接通延时定时器时序图如图 4-24 所示。

（9）SF　　断开延时定时器

指令格式为：SF <定时器>

<定时器>的数据类型为 TIMER，存储区为 T，定时器编号、范围取决于 CPU。SF <定时器> 在 RLO 从“1”跳转到“0”时起动寻址的定时器。只要 RLO = 1，程序时间间隔就会流逝。

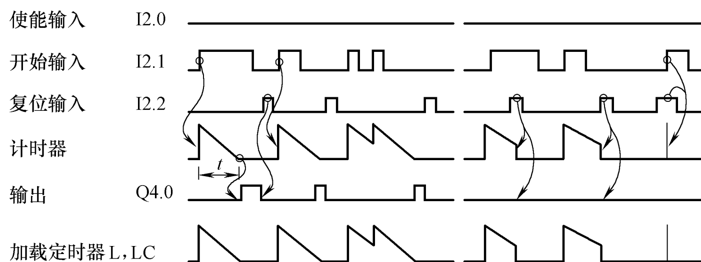


图 4-24 带保持的接通延时定时器时序图

如果在程序时间间隔截止之前 RLO 跳转到“1”，则停止计时。此定时器起动指令要求将时间值和时间基准作为 BCD 数存储在 ACCU 1-L 中。

例：

```
A      I 2.0
FR     T1           //起用定时器 T1
A      I2.1
L    S5T#10s        //在 ACCU 1 中预置 10s
SF     T1           //将定时器 T1 起动为断开延时定时器
A      I2.2
R      T1           //复位定时器 T1
A      T1           //检查定时器 T1 的信号状态
 =    Q4.0
L      T1           //以二进制形式装载定时器 T1 的当前定时器值
T      MW10
LC     T1           //以 BCD 形式装载定时器 T1 的当前定时器值
T      MW12
```

断开延时定时器时序图如图 4-25 所示。

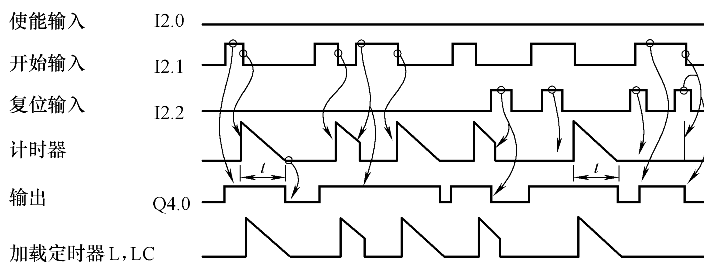
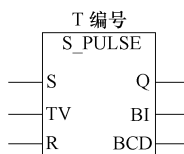


图 4-25 断开延时定时器时序图

2. 梯形图（LAD）的定时器指令

(1) S_PULSE 脉冲 S5 定时器

指令符号为：



T 编号的数据类型为 TIMER，存储区为 T；S（使能输入）、R（复位输入）和 Q（定时器的状态）的数据类型为布尔型，存储区为 I、Q、M、L、D；TV（预设时间值）的数据类型为 S5TIME，存储区为 I、Q、M、L、D；BI（剩余时间值，整型格式）的数据类型为字，存储区为 I、Q、M、L、D；BCD（剩余时间值，BCD 格式）的数据类型为字，存储区为 I、Q、M、L、D。

如果在起动（S）输入端有一个上升沿，S_PULSE（脉冲 S5 定时器）将起动指定的定时器。信号变化始终是起用定时器的必要条件。定时器在输入端 S 的信号状态为“1”时运行，但最长周期是由输入端 TV 指定的时间值。只要定时器运行，输出端 Q 的信号状态就为“1”。如果在时间间隔结束前，S 输入端从“1”变为“0”，则定时器将停止。这种情况下，输出端 Q 的信号状态为“0”。

如果在定时器运行期间定时器复位（R）输入从“0”变为“1”时，则定时器将被复位。当前时间和时间基准也被设置为零。如果定时器不是正在运行，则定时器 R 输入端的逻辑“1”没有任何作用。

可在输出端 BI 和 BCD 上扫描当前时间值。时间值在 BI 端是二进制编码，在 BCD 端是 BCD 编码。当前时间值为初始 TV 值减去定时器起动后经过的时间。

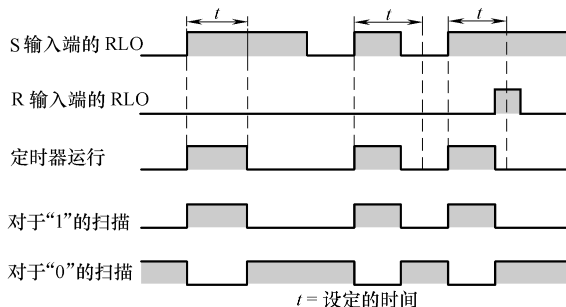
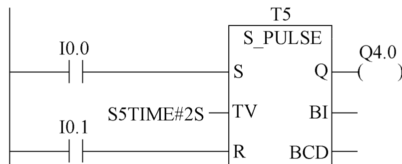


图 4-26 脉冲定时器特性曲线

脉冲定时器特性曲线如图 4-26 所示。

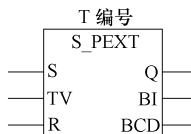
例：



如果输入端 I0.0 的信号状态从“0”变为“1”（RLO 中的上升沿），则定时器 T5 将起动。只要 I0.0 为“1”，定时器就将继续运行指定的 2s 时间。如果定时器达到预定时间前，I0.0 的信号状态从“1”变为“0”，则定时器将停止。如果输入端 I0.1 的信号状态从“0”变为“1”，而定时器仍在运行，则时间复位。只要定时器运行，输出端 Q4.0 就是逻辑“1”，如果定时器预设时间结束或复位，则输出端 Q4.0 变为“0”。

（2）S_PEXT 扩展脉冲 S5 定时器

指令符号为：



T 编号的数据类型为 TIMER，存储区为 T；S（使能输入）、R（复位输入）和 Q（定时器的状态）的数据类型为布尔型，存储区为 I、Q、M、L、D；TV（预设时间值）的数据类型为 S5TIME，存储区为 I、Q、M、L、D；BI（剩余时间值，整型格式）的数据类型为字，存储区为 I、Q、M、L、D；BCD（剩余时间值，BCD 格式）的数据类型为字，存储区为 I、Q、M、L、D。

如果在起动（S）输入端有一个上升沿，S_PEXT（扩展脉冲 S5 定时器）将起动指定的定时器。信号变化始终是起用定时器的必要条件。定时器以在输入端 TV 指定的预设时间间隔运行，即使在时间间隔结束前，S 输入端的信号状态变为“0”。只要定时器运行，输出端 Q 的信号状态

就为“1”。如果在定时器运行期间输入端 S 的信号状态从“0”变为“1”，则将使用预设的时间值重新起动（“重新触发”）定时器。

如果在定时器运行期间复位（R）输入端从“0”变为“1”，则定时器复位。当前时间和时间基准被设置为零。可在输出端 BI 和 BCD 扫描当前时间值。时间值在 BI 处为二进制编码，在 BCD 处为 BCD 编码。当前时间值为初始 TV 值减去定时器起动后经过的时间。

扩展脉冲定时器特征曲线如图 4-27 所示。

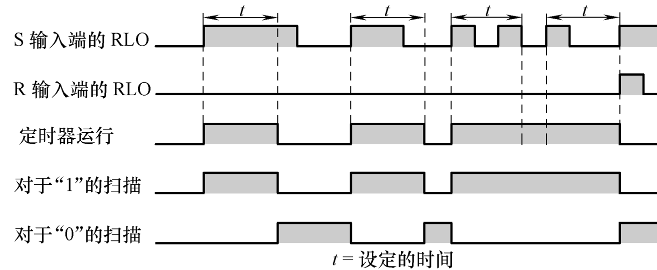
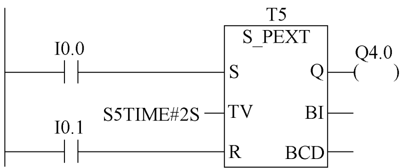


图 4-27　扩展脉冲定时器特征曲线

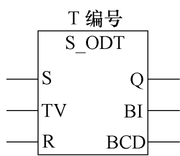
例：

如果输入端 I0.0 的信号状态从“0”变为“1”（RLO 中的上升沿），则定时器 T5 将起动。定时器将继续运行指定的 2s 时间，而不会受到输入端 S 处下降沿的影响。如果在定时器达到预定时间前，I0.0 的信号状态从“0”变为“1”，则定时器将被重新触发。只要定时器运行，输出端 Q4.0 就为逻辑“1”。



（3）S_ODT 接通延时 S5 定时器

指令符号为：



T 编号的数据类型为 TIMER，存储区为 T；S（使能输入）、R（复位输入）和 Q（定时器的状态）的数据类型为布尔型，存储区为 I、Q、M、L、D；TV（预设时间值）的数据类型为 S5TIME，存储区为 I、Q、M、L、D；BI（剩余时间值，整型格式）的数据类型为字，存储区为 I、Q、M、L、D；BCD（剩余时间值，BCD 格式）的数据类型为字，存储区为 I、Q、M、L、D。

如果在起动（S）输入端有一个上升沿，S_ODT（接通延时 S5 定时器）将起动指定的定时器。信号变化始终是起用定时器的必要条件。只要输入端 S 的信号状态为正，定时器就以在输入端 TV 指定的时间间隔运行。定时器达到指定时间而没有出错，并且 S 输入端的信号状态仍为“1”时，输出端 Q 的信号状态为“1”。如果定时器运行期间输入端 S 的信号状态从“1”变为“0”，定时器将停止。这种情况下，输出端 Q 的信号状态为“0”。

如果在定时器运行期间复位（R）输入从“0”变为“1”，则定时器复位。当前时间和时间基准被设置为零。然后，输出端 Q 的信号状态变为“0”。如果在定时器没有运行时 R 输入端有一个逻辑“1”，并且输入端 S 的 RLO 为“1”，则定时器也复位。可在输出端 BI 和 BCD 扫描当前时间值。时间值在 BI 处为二进制编码，在 BCD 处为 BCD 编码。当前时间值为初始 TV 值减去定时器起动后经过的时间。

接通延时定时器特征曲线如图 4-28 所示。

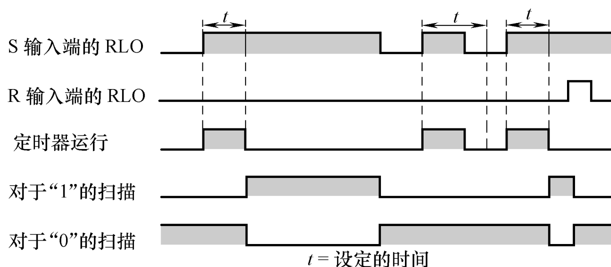
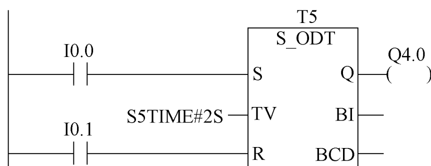


图 4-28　接通延时定时器特征曲线

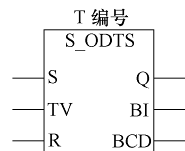
例：

如果 I0.0 的信号状态从“0”变为“1”（RLO 中的上升沿），则定时器 T5 将起动。如果指定的 2s 时间结束并且输入端 I0.0 的信号状态仍为“1”，则输出端 Q4.0 将为“1”。如果 I0.0 的信号状态从“1”变为“0”，则定时器停止，并且 Q4.0 将为“0”（如果 I0.1 的信号状态从“0”变为“1”，则无论定时器是否运行，时间都复位）。



（4）S_ODTS 保持接通延时 S5 定时器

指令符号为：



T 编号的数据类型为 TIMER，存储区为 T；S（使能输入）、R（复位输入）和 Q（定时器的状态）的数据类型为布尔型，存储区为 I、Q、M、L、D；TV（预设时间值）的数据类型为 S5TIME，存储区为 I、Q、M、L、D；BI（剩余时间值，整型格式）的数据类型为字，存储区为 I、Q、M、L、D；BCD（剩余时间值，BCD 格式）的数据类型为字，存储区为 I、Q、M、L、D。

如果在起动（S）输入端有一个上升沿，S_ODTS（保持接通延时 S5 定时器）将起动指定的定时器。信号变化始终是起用定时器的必要条件。定时器以在输入端 TV 指定的时间间隔运行，即使在时间间隔结束前，输入端 S 的信号状态变为“0”。定时器预定时间结束时，输出端 Q 的信号状态为“1”，而无论输入端 S 的信号状态如何。如果在定时器运行时输入端 S 的信号状态从“0”变为“1”，则定时器将以指定的时间重新起动（重新触发）。

如果复位（R）输入从“0”变为“1”，则无论 S 输入端的 RLO 如何，定时器都将复位。然后，输出端 Q 的信号状态变为“0”。可在输出端 BI 和 BCD 扫描当前时间值。时间值在 BI 端是二进制编码，在 BCD 端是 BCD 编码。当前时间值为初始 TV 值减去定时器起动后经过的时间。

保持接通延时定时器特征曲线如图 4-29 所示。

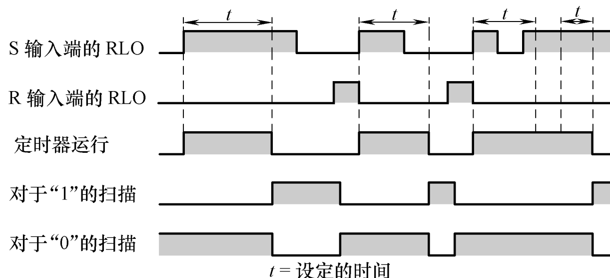
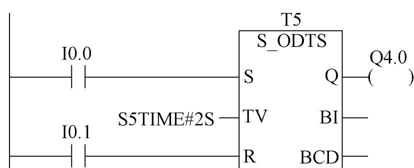


图 4-29　保持接通延时定时器特征曲线

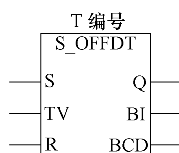
例：

如果 I0.0 的信号状态从“0”变为“1”（RLO 中的上升沿），则定时器 T5 将起动。无论 I0.0 的信号是否从“1”变为“0”，定时器都将运行。如果在定时器达到指定时间前，I0.0 的信号状态从“0”变为“1”，则定时器将重新触发。如果定时器达到指定时间，则输出端 Q4.0 将变为“1”（如果输入端 I0.1 的信号状态从“0”变为“1”，则无论 S 处的 RLO 如何，时间都将复位）。



（5）S_OFFDT 断开延时 S5 定时器

指令符号为：



T 编号的数据类型为 TIMER，存储区为 T；S（使能输入）、R（复位输入）和 Q（定时器的状态）的数据类型为布尔型，存储区为 I、Q、M、L、D；TV（预设时间值）的数据类型为 S5TIME，存储区为 I、Q、M、L、D；BI（剩余时间值，整型格式）的数据类型为字，存储区为 I、Q、M、L、D；BCD（剩余时间值，BCD 格式）的数据类型为字，存储区为 I、Q、M、L、D。

如果在起动（S）输入端有一个下降沿，S_OFFDT（断开延时 S5 定时器）将起动指定的定时器。信号变化始终是起用定时器的必要条件。如果 S 输入端的信号状态为“1”，或定时器正在运行，则输出端 Q 的信号状态为“1”。如果在定时器运行期间输入端 S 的信号状态从“0”变为“1”时，定时器将复位。输入端 S 的信号状态再次从“1”变为“0”后，定时器才能重新起动。

如果在定时器运行期间复位（R）输入从“0”变为“1”时，定时器将复位。

可在输出端 BI 和 BCD 扫描当前时间值。时间值在 BI 端是二进制编码，在 BCD 端是 BCD 编码。当前时间值为初始 TV 值减去定时器起动后经过的时间。

断开延时定时器特征曲线如图 4-30 所示。

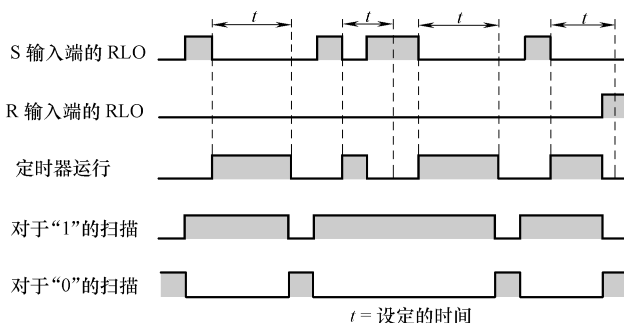
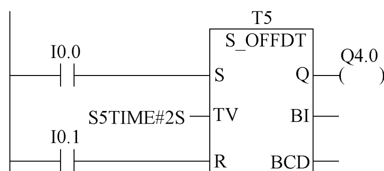


图 4-30　断开延时定时器特征曲线

例：

如果 I0.0 的信号状态从“1”变为“0”，则定时器起动。

I0.0 为“1”或定时器运行时，Q4.0 为“1”（如果在定时器运行期间 I0.1 的信号状态从“0”变为“1”，则定时器复位）。



（6）---(SP) 脉冲定时器线圈

指令符号为：

<T 编号>

---(SP)

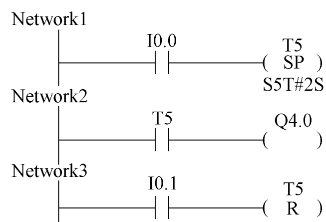
<时间值>

T 编号的数据类型为 TIMER,存储区为 T；<时间值>（预设时间值）的数据类型为 S5TIME，存储区为 I、Q、M、L、D。

如果 RLO 状态有一个上升沿，---(SP)（脉冲定时器线圈）将以该<时间值>起动指定的定时器。只要 RLO 保持正值（“1”），定时器就继续运行指定的时间间隔。只要定时器运行，计数器的信号状态就为“1”。如果在达到时间值前，RLO 中的信号状态从“1”变为“0”，则定时器将停止。这种情况下，对于“1”的扫描始终产生结果“0”。

例：

如果输入端 I0.0 的信号状态从“0”变为“1”（RLO 中的上升沿），则定时器 T5 起动。只要输入端 I0.0 的信号状态为“1”，定时器就继续运行指定的 2s 时间。如果在指定的时间结束前输入端 I0.0 的信号状态从“1”变为“0”，则定时器停止。只要定时器运行，输出端 Q4.0 的信号状态就为“1”。如果输入端 I0.1 的信号状态从“0”变为“1”，定时器 T5 将复位，定时器停止，并将时间值的剩余部分清“0”。



（7）---(SE) 扩展脉冲定时器线圈

指令符号为：

<T 编号>

---(SE)

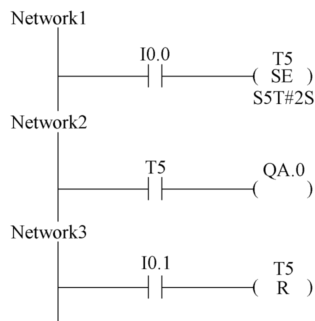
<时间值>

T 编号的数据类型为 TIMER,存储区为 T；<时间值>（预设时间值）的数据类型为 S5TIME，存储区为 I、Q、M、L、D。

如果 RLO 状态有一个上升沿，---(SE)（扩展脉冲定时器线圈）将以指定的<时间值>起动指定的定时器。定时器继续运行指定的时间间隔，即使定时器达到指定时间前 RLO 变为“0”。只要定时器运行，计数器的信号状态就为“1”。如果在定时器运行期间 RLO 从“0”变为“1”，则将以指定的时间值重新起动定时器（重新触发）。

例：

如果输入端 I0.0 的信号状态从“0”变为“1”（RLO 中的上升沿），则定时器 T5 起动。定时器继续运行，而无论 RLO 是否出现下降沿。如果在定时器达到指定时间前 I0.0 的信号状态从“0”变为“1”，则定时器重新触发。只要定时器运行，输出端 Q4.0 的信号状态就为“1”。如果输入端 I0.1 的信号状态从“0”变为“1”，定时器 T5 将复位，定时器停止，并将时间值的剩余部分清为“0”。



（8）---(SD) 接通延时定时器线圈

指令符号为：

<T 编号>

---(SD)

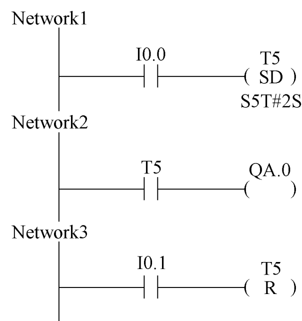
<时间值>

T 编号的数据类型为 TIMER,存储区为 T；<时间值>（预设时间值）的数据类型为 S5TIME，存储区为 I、Q、M、L、D。

如果 RLO 状态有一个上升沿，---(SD)（接通延时定时器线圈）将以该<时间值>起动指定的定时器。如果达到该<时间值>而没有出错，且 RLO 仍为“1”，则定时器的信号状态为“1”。如果在定时器运行期间 RLO 从“1”变为“0”，则定时器复位。这种情况下，对于“1”的扫描始终产生结果“0”。

例：

如果输入端 I0.0 的信号状态从“0”变为“1”（RLO 中的上升沿），则定时器 T5 起动。如果指定时间结束而输入端 I0.0 的信号状态仍为“1”，则输出端 Q4.0 的信号状态将为“1”。如果输入端 I0.0 的信号状态从“1”变为“0”，则定时器保持空闲，并且输出端 Q4.0 的信号状态将为“0”。如果输入端 I0.1 的信号状态从“0”变为“1”，定时器 T5 将复位，定时器停止，并将时间值的剩余部分清为“0”。



（9）---(SS) 保持接通延时定时器线圈

指令符号为：

<T 编号>

---(SS)

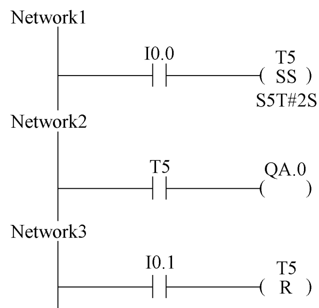
<时间值>

T 编号的数据类型为 TIMER,存储区为 T；<时间值>（预设时间值）的数据类型为 S5TIME，存储区为 I、Q、M、L、D。

如果 RLO 状态有一个上升沿，---(SS)（保持接通延时定时器线圈）将起动指定的定时器。如果达到时间值，定时器的信号状态为“1”。只有明确进行复位，定时器才可能重新起动。只有复位才能将定时器的信号状态设为“0”。如果在定时器运行期间 RLO 从“0”变为“1”，则定时器以指定的时间值重新起动。

例：

如果输入端 I0.0 的信号状态从“0”变为“1”（RLO 中的上升沿），则定时器 T5 起动。如果在定时器达到指定时间前输入端 I0.0 的信号状态从“0”变为“1”，则定时器将重新触发。如果定时器达到指定时间，则输出端 Q4.0 将变为“1”。输入端 I0.1 的信号状态“1”将复位定时器 T5，使定时器停止，并将时间值的剩余部分清为“0”。



（10）---(SF) 断开延时定时器线圈

指令符号为：

<T 编号>

---(SF)

<时间值>

T 编号的数据类型为 TIMER,存储区为 T；<时间值>（预设时间值）的数据类型为 S5TIME，存储区为 I、Q、M、L、D。

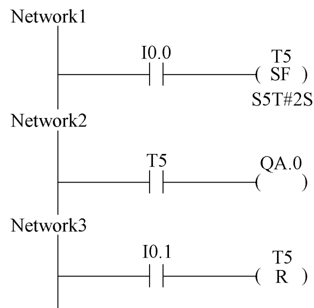
如果 RLO 状态有一个下降沿，---(SF)（断开延时定时器线圈）将起动指定的定时器。当

RLO 为“1”时或只要定时器在 < 时间值 > 时间间隔内运行，定时器就为“1”。如果在定时器运行期间 RLO 从“0”变为“1”，则定时器复位。只要 RLO 从“1”变为“0”，定时器即会重新起动。

例：

如果输入端 I0.0 的信号状态从“1”变为“0”，则定时器起动。

如果输入端 I0.0 为“1”或定时器正在运行，则输出端 Q4.0 的信号状态为“1”。如果输入端 I0.1 的信号状态从“0”变为“1”，定时器 T5 将复位，定时器停止，并将时间值的剩余部分清为“0”。



十四、字逻辑指令

字逻辑指令按照布尔逻辑逐位比较字（16 位）和双字（32 位）。每个字或双字必须位于两个累加器其中一个之内。

对于字而言，累加器 2 的低字中的内容会与累加器 1 的低字中的内容组合。组合结果存储在累加器 1 的低字中，同时覆盖原有的内容。对于双字而言，累加器 2 的内容与累加器 1 的内容相组合。组合结果存储在累加器 1 中，同时覆盖原有的内容。如结果不等于 0，则将状态字的位 CC 1 置为“1”。如结果等于 0，则将状态字的位 CC 1 置为“0”。

1. 语句表（STL）的字逻辑指令

（1）AW　单字与运算（16 位）

指令格式为：AW

AW < 常数 >

< 常数 > 的数据类型为 WORD，16 位常数，使用与运算（AND）同 ACCU 1-L 组合的位模式。

AW（单字与运算）根据布尔逻辑与运算，将 ACCU 1-L 的内容与 ACCU 2-L 的内容或与 16 位常数逐位进行组合。只有当两个在逻辑运算中组合的两个字的相应位都为“1”时，结果字中的位才为“1”。结果存储在 ACCU 1-L 中。ACCU 1-H 和 ACCU 2（以及 ACCU 3 和 ACCU 4，对于具有 4 个 ACCU 的 CPU）保持不变。状态位 CC 1 被置为运算结果（如结果不等于零，则 CC 1 = 1）。复位状态字位 CC 0 和 OV 为 0。

例 1：

```
L     IW20          //将 IW20 的内容装入 ACCU 1-L
L     IW22          //将 ACCU 1 的内容装入 ACCU 2 中。将 IW22 的内容装入 ACCU 1-L
AW                  //使用与运算将 ACCU 1-L 中的位与 ACCU 2-L 的位相组合，将结果存
                      储在 ACCU 1-L 中
T     MW 8          //将结果传送到 MW8
```

例 2：

```
L     IW20          //将 IW20 的内容装入 ACCU 1-L
AW    W#16#0FFF     //使用与运算将 ACCU 1-L 的位与 16 位常数（0000_1111_1111_1111）
                      的位模式组合，将结果存储在 ACCU 1-L 中
JP    NEXT          //如结果不等于零，则跳转到下一个跳转标签（CC 1 = 1）
```

（2）OW　单字或运算（16 位）

指令格式为：OW

OW < 常数 >

<常数>的数据类型为 WORD,16 位常数，使用或运算（OR）与 ACCU 1-L 组合的位模式。

OW（单字或运算）根据布尔逻辑或运算，将 ACCU 1-L 的内容与 ACCU 2-L 的内容或与 16 位常数逐位组合。只有当在逻辑运算中组合的两个字的相应位中至少有一位是“1”时，结果字中的位才为“1”。结果存储在 ACCU 1-L 中。ACCU 1-H 和 ACCU 2（以及 ACCU 3 和 ACCU 4，对于具有 4 个 ACCU 的 CPU）保持不变。执行指令时既不考虑也不会影响 RLO。状态位 CC 1 被置为运算的结果（如结果不等于零，则 CC 1 =1）。复位状态字位 CC 0 和 OV 为 0。

例 1：

```
L     IW20          //将 IW20 的内容装入 ACCU 1-L
L     IW22          //将 ACCU 1 的内容装入 ACCU 2。将 IW22 的内容装入 ACCU 1-L
OW                  //使用或运算，将 ACCU 1-L 中的位与 ACCU 2-L 的位组合，将结果存
                      储在 ACCU 1-L 中
T     MW8           //将结果传送到 MW8
```

例 2：

```
L     IW20          //将 IW 20 的内容装入 ACCU 1-L
OW    W#16#0FFF     //使用或运算，将 ACCU 1-L 的位与 16 位常数（0000_1111_1111_1111）
                      的位模式相组合，将结果存储在 ACCU 1-L 中
JP    NEXT          //如结果不等于零，则跳转到下一个跳转标签（CC 1 =1）
```

（3）XOW　单字异或运算（16 位）

指令格式为：XOW

XOW <常数>

<常数>的数据类型为 WORD,16 位常数，使用异或（XOR）运算与 ACCU 1-L 组合的位模式。

XOW（单字异或运算）根据布尔逻辑异或运算，将 ACCU 1-L 的内容与 ACCU 2-L 的内容或与 16 位的常数逐位组合。只有当两个在逻辑运算中组合的字的相应位中有一位是“1”时，结果字中的位才为“1”。结果存储在 ACCU 1-L 中。ACCU 1-H 和 ACCU 2 保持不变。状态位 CC 1 被置为运算的结果（如结果不等于零，则 CC 1 =1）。复位状态字位 CC 0 和 OV 为 0。可多次使用异或运算函数。如果所选中地址有奇数个“1”，则逻辑运算结果为“1”。

例 1：

```
L     IW20          //将 IW20 的内容装入 ACCU 1-L
L     IW22          //将 ACCU 1 的内容装入 ACCU 2。将 ID24 的内容装入 ACCU 1-L
XOW                 //使用异或运算，将 ACCU 1-L 的位与 ACCU 2-L 的位组合，将结果存
                      储在 ACCU 1-L 中
T     MW8           //将结果传送到 MW8
```

例 2：

```
L     IW20          //将 IW20 的内容装载到 ACCU 1-L 中
XOW   16#0FFF       //使用异或运算，将 ACCU 1-L 的位与 16 位常数（0000_1111_1111_
                      1111）的位模式相组合，将结果存储在 ACCU 1-L 中
JP    NEXT          //如结果不等于零，则跳转到下一个跳转标签（CC 1 =1）
```

（4）AD　双字与运算（32 位）

指令格式为：AD

AD <常数>

<常数>的数据类型为 DWORD,32 位常数，使用与运算（AND）与 ACCU 1 组合的位模式。

AD（双字与运算）根据布尔逻辑与运算，将 ACCU 1 的内容与 ACCU 2 的内容或与 32 位的常数逐位进行组合。只有当在逻辑运算中组合的两个双字的相应位都为“1”时，结果双字中的位才为“1”。该结果存储在 ACCU 1 中。ACCU 2（以及 ACCU 3 和 ACCU 4，对于具有 4 个 ACCU 的 CPU）保持不变。状态位 CC 1 被置为运算结果（如结果不等于零，则 CC 1 =1）。复位状态字位 CC 0 和 OV 为 0。

例 1：

```
L    ID20              //将 ID20 的内容装入 ACCU 1
L    ID24              //将 ACCU 1 的内容装入 ACCU 2。将 ID24 的内容装入 ACCU 1
AD                     //使用与运算将 ACCU 1 中的位与 ACCU 2 中的位组合，将结果
                         存入 ACCU 1 中
T    MD8               //将结果传送到 MD8
```

例 2：

```
L    ID 20             //将 ID20 的内容装入 ACCU 1
AD   DW#16#0FFF_EF21   //使用与将（0000_1111_1111_1111_1110_1111_0010_0001）与
                         ACCU 1 的位模式组合，结果存入 ACCU 1 中
JP   NEXT              //如结果不等于零，则跳转到下一个跳转标签（CC 1 =1）
```

（5）OD　双字或运算（32 位）

指令格式为：OD

OD <常数>

<常数>的数据类型为 DWORD,32 位常数，使用或运算（OR）与 ACCU 1 组合的位模式。

OD（双字或运算）根据布尔逻辑或运算，将 ACCU 1 的内容与 ACCU 2 的内容或与 32 位常数逐位组合。只有当在逻辑运算中组合的两个双字的相应位中至少有一位是“1”时，结果双字中的位才为“1”。结果存储在 ACCU 1 中。ACCU 2（以及 ACCU 3 和 ACCU 4，对于具有 4 个 ACCU 的 CPU）保持不变。状态位 CC 1 被置为运算结果的函数（如结果不等于零，则 CC 1 =1）。复位状态字位 CC 0 和 OV 为 0。

例 1：

```
L    ID20              //将 ID20 的内容装入 ACCU 1
L    ID24              //将 ACCU 1 的内容装入 ACCU 2。将 ID24 的内容装入 ACCU 1
OD                     //使用或运算，将 ACCU 1 中的位与 ACCU 2 中的位组合，将结
                         果存入 ACCU 1
T    MD8               //将结果传送到 MD8
```

例 2：

```
L    ID20              //将 ID20 的内容装入 ACCU 1
OD   DW#16#0FFF_EF21   //使用或运算，将 ACCU 1 中的位与 32 位常数的位模式组合，
                         结果存入 ACCU 1
JP   NEXT              //如结果不等于零，则跳转到下一个跳转标签（CC 1 =1）
```

（6）XOD　双字异或运算（32 位）

指令格式为：XOD

XOD <常数>

<常数>的数据类型为 DWORD,32 位常数，使用异或（XOR）运算与 ACCU 1 组合的位

模式。

XOD（双字异或运算）根据布尔逻辑 XOR（异或）运算，将 ACCU 1 的内容与 ACCU 2 的内容或与 32 位常数逐位组合。当在逻辑运算中组合的两个双字的相应位中仅有一位是“1”时，结果双字中的位才为“1”。结果存储在 ACCU 1 中。ACCU 2 保持不变。状态位 CC 1 被置为运算的结果（如结果不等于零，则 CC 1 = 1）。复位状态字位 CC 0 和 OV 为 0。可多次使用异或运算函数。如果选中地址有奇数个“1”，则逻辑运算结果为“1”。

例 1：

```
L      ID20                 //将 ID20 的内容装入 ACCU 1
L      ID24                 //将 ACCU 1 的内容装入 ACCU 2。将 ID24 的内容装入 ACCU 1
XOD                         //使用异或运算，将 ACCU 1 中的位与 ACCU 2 的位组合，将结
                              果存入 ACCU 1
T      MD8                  //将结果传送到 MD8
```

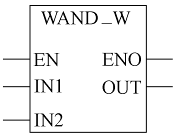
例 2：

```
L      ID20                 //将 ID20 的内容装入 ACCU 1
XOD    DW#16#0FFF_EF21      //使用异或运算，将 ACCU 1 中的位与 32 位常数的位模式组
                              合，结果存入 ACCU 1
JP     NEXT                 //如结果不等于零，则跳转到下一个跳转标签（CC 1 = 1）
```

2. 梯形图（LAD）的字逻辑指令

（1）WAND_W　（字）单字与运算

指令符号为：

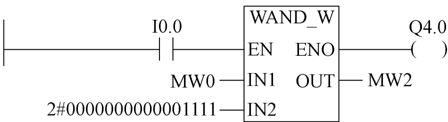


EN（使能输入）和 ENO（使能输出）的数据类型为 BOOL，存储器区为 I、Q、M、L、D；IN1（逻辑运算的第一个值）、IN2（逻辑运算的第二个值）和 OUT（逻辑运算的结果字）的数据类型为 WORD，存储器区为 I、Q、M、L、D。

使能（EN）输入的信号状态为“1”时将激活 WAND_W（字与运算），并逐位对 IN1 和 IN2 处的两个字值进行与运算。按纯位模式来解释这些值。可以在输出 OUT 处扫描结果。ENO 与 EN 的逻辑状态相同。

例：

如果 I0.0 为“1”，则执行指令。在 MW0 的位中，只有 0 ~ 3 位是相关的，其余位被 IN2 字位模式屏蔽。



MW0　=　01010101 01010101

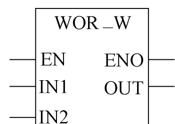
IN2　=　00000000 00001111

MW0 AND IN2 = MW2　=　00000000 00000101

如果执行了指令，则 Q4.0 为“1”。

（2）WOR_W　（字）单字或运算

指令符号为：

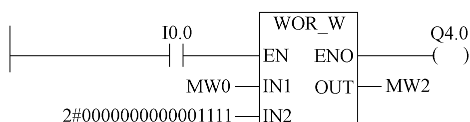


EN（使能输入）和 ENO（使能输出）的数据类型为 BOOL，存储器区为 I、Q、M、L、D；IN1（逻辑运算的第一个值）、IN2 （逻辑运算的第二个值）和 OUT（逻辑运算的结果字）的数据类型为 WORD，存储器区为 I、Q、M、L、D。

使能（EN）输入的信号状态为“1”时将激活 WOR_W（单字或运算），并逐位对 IN1 和 IN2 处的两个字值进行或运算。按纯位模式来解释这些值。可以在输出 OUT 处扫描结果。ENO 与 EN 的逻辑状态相同。

例：

如果 I0.0 为“1”，则执行指令。将位 0 ~ 3 设置为“1”，不改变 MW0 的所有其他位。



MW0 = 01010101 01010101

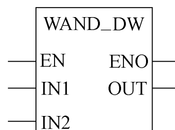
IN2 = 00000000 00001111

MW0 OR IN2 = MW2 = 01010101 01011111

如果执行了指令，则 Q4.0 为“1”。

（3）WAND_DW （字）双字与运算

指令符号为：

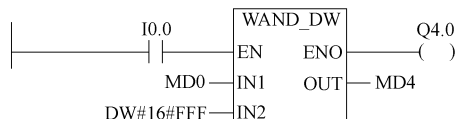


EN（使能输入）和 ENO（使能输出）的数据类型为 BOOL，存储器区为 I、Q、M、L、D；IN1（逻辑运算的第一个值）、IN2（逻辑运算的第二个值）和 OUT（逻辑运算的结果双字）的数据类型为 DWORD，存储器区为 I、Q、M、L、D。

使能（EN）输入的信号状态为“1”时将激活 WAND_DW（双字与运算），并逐位对 IN1 和 IN2 处的两个字值进行与运算。按纯位模式来解释这些值。可以在输出 OUT 处扫描结果。ENO 与 EN 的逻辑状态相同。

例：

如果 I0.0 为“1”，则执行指令。在 MD0 的位中，只有 0 和 11 位是相关的，其余位被 IN2 位模式屏蔽。



MD0 = 01010101 01010101 01010101 01010101

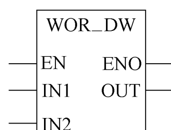
IN2 = 00000000 00000000 00001111 11111111

MD0 AND IN2 = MD4 = 00000000 00000000 00000101 01010101

如果执行了指令，则 Q4.0 为“1”。

（4）WOR_DW （字）双字或运算

指令符号为：



EN（使能输入）和 ENO（使能输出）的数据类型为 BOOL，存储器区为 I、Q、M、L、D；IN1（逻辑运算的第一个值）、IN2（逻辑运算的第二个值）和 OUT（逻辑运算的结果双字）的数据类型为 DWORD，存储器区为 I、Q、M、L、D。

使能（EN）输入的信号状态为“1”时将激活 WOR_DW（双字或运算），并逐位对 IN1 和 IN2 处的两个字值进行或运算。按纯位模式来解释这些值。可以在输出 OUT 处扫描结果。ENO 与 EN 的逻辑状态相同。

例：

如果 I0.0 为“1”，则执行指令。将位 0～11 设置为“1”，不改变 MD0 的其余位：



MD0 = 01010101 01010101 01010101 01010101

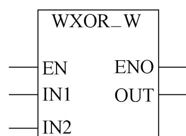
IN2 = 00000000 00000000 00001111 11111111

MD0 OR IN2 = MD4 = 01010101 01010101 01011111 11111111

如果执行了指令，则 Q4.0 为“1”。

(5) WXOR_W （字）单字异或运算

指令符号为：



EN（使能输入）和 ENO（使能输出）的数据类型为 BOOL，存储器区为 I、Q、M、L、D；IN1（逻辑运算的第一个值）、IN2 （逻辑运算的第二个值）和 OUT（逻辑运算的结果字）的数据类型为 WORD，存储器区为 I、Q、M、L、D。

使能（EN）输入的信号状态为“1”时将激活 WXOR_W（单字异或运算），并逐位对 IN1 和 IN2 处的两个字值进行异或运算。按纯位模式来解释这些值。可以在输出 OUT 处扫描结果。ENO 与 EN 的逻辑状态相同。

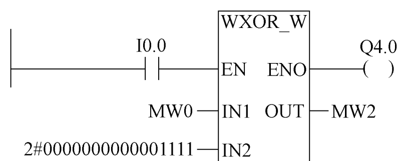
例：

如果 I0.0 为“1”，则执行指令：

MW0 = 01010101 01010101

IN2 = 00000000 00001111

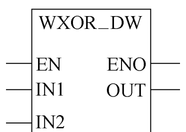
MW0 XOR IN2 = MW2 = 01010101 01011010



如果执行了指令，则 Q4.0 为“1”。

(6) WXOR_DW （字）双字异或运算

指令符号为：



EN（使能输入）和 ENO（使能输出）的数据类型为 BOOL，存储器区为 I、Q、M、L、D；IN1（逻辑运算的第一个值）、IN2 （逻辑运算的第二个值）和 OUT（逻辑运算的结果双字）的数据类型为 DWORD，存储器区为 I、Q、M、L、D。

使能（EN）输入的信号状态为“1”时将激活 WXOR_DW（双字异或运算），并逐位对 IN1 和 IN2 处的两个字值进行异或运算。按纯位模式来解释这些值。可以在输出 OUT 扫描结果。ENO 与 EN 的逻辑状态相同。

例：

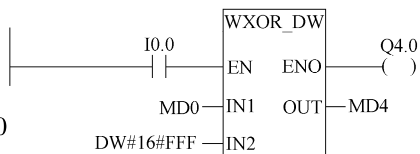
如果 I0.0 为“1”，则执行指令：

MD0 = 01010101 01010101 01010101 01010101

IN2 = 00000000 00000000 00001111 11111111

MW2 = MD0 XOR IN2 = 01010101010101010101011010 10101010

如果执行了指令，则 Q4.0 为“1”。



十五、累加器（STL）指令

累加器指令用于处理一个或两个累加器的内容。

（1）TAK　将 ACCU 1 与 ACCU 2 互换

指令格式为：TAK

TAK（将 ACCU 1 与 ACCU 2 互换）将把 ACCU 1 的内容与 ACCU 2 的内容交换。执行该指令时不考虑状态位，也不会影响状态位。对具有 4 个 ACCU 的 CPU，ACCU 3 和 ACCU 4 的内容保持不变。

例：从较大值中减去较小值

```
L     MW10        //将 MW10 的内容载入 ACCU 1-L
L     MW12        //将 ACCU 1-L 的内容载入 ACCU 2-L。将 MW12 的内容载入
                    ACCU 1-L
>I                //检查 ACCU 2-L（MW10）是否大于 ACCU 1-L（MW12）
JC    NEXT        //如果 ACCU 2（MW10）大于 ACCU 1（MW12），则跳转到
                    NEXT 跳转标签
TAK               //将 ACCU 1 的内容与 ACCU 2 的内容交换
NEXT:-I           //从 ACCU 1-L 的内容中减去 ACCU 2-L 的内容
T     MW14        //将结果（= 较大值减较小值）传送到 MW14
```

（2）POP　具有两个 ACCU 的 CPU

指令格式为：POP

POP（具有两个 ACCU 的 CPU）将 ACCU 2 的全部内容复制到 ACCU 1。ACCU 2 保持不变。执行该指令时不考虑状态位，也不会影响状态位。

例：

```
T     MD10        //将 ACCU 1 的内容（= 值 A）传送到 MD10
POP               //将 ACCU 2 的整个内容复制到 ACCU 1
T     MD14        //将 ACCU 1 的内容（= 值 B）传送到 MD14
```

（3）POP　具有 4 个 ACCU 的 CPU

指令格式为：POP

POP（具有 4 个 ACCU 的 CPU）将 ACCU 2 的全部内容复制到 ACCU 1，将 ACCU 3 的内容复制到 ACCU 2，并将 ACCU 4 的内容复制到 ACCU 3。ACCU 4 保持不变。执行该指令时不考虑状

态位，也不会影响状态位。

目录　　　　　　　　　ACCU 1　ACCU 2　ACCU 3　ACCU 4
执行 POP 指令之前　值 A　　　值 B　　　值 C　　　值 D
执行 POP 指令之后　值 B　　　值 C　　　值 D　　　值 D

例：

```
T    MD10          //将 ACCU 1 的内容（＝值 A）传送到 MD10
POP                //将 ACCU 2 的整个内容复制到 ACCU 1
T    MD14          //将 ACCU 1 的内容（＝值 B）传送到 MD14
```

（4）PUSH　具有两个 ACCU 的 CPU

指令格式为：PUSH

PUSH（ACCU 1 到 ACCU 2）将 ACCU 1 的整个内容复制到 ACCU 2。ACCU 1 保持不变。执行该指令时不考虑状态位，也不会影响状态位。

例：

```
L    MW10          //将 MW10 的内容载入 ACCU 1
PUSH               //将 ACCU 1 的整个内容复制到 ACCU 2
```

目录　　　　　　　　ACCU 1　　　ACCU 2
执行 PUSH 指令之前　　<MW10>　　　<X>
执行 PUSH 指令之后　　<MW10>　　<MW10>

（5）PUSH　具有 4 个 ACCU 的 CPU

指令格式为：PUSH

PUSH（具有 4 个 ACCU 的 CPU）将 ACCU 3 的内容复制到 ACCU 4，将 ACCU 2 的内容复制到 ACCU 3，并将 ACCU 1 的内容复制到 ACCU 2。ACCU 1 保持不变。执行该指令时不考虑状态位，也不会影响状态位。

例：

```
L    MW10          //将 MW10 的内容载入 ACCU 1
PUSH               //将 ACCU 1 的整个内容复制到 ACCU 2，将 ACCU 2 的内容复制到 ACCU 3，并将 ACCU 3 的内容复制到 ACCU 4
```

目录　　　　　　　　ACCU 1　ACCU 2　ACCU 3　ACCU 4
执行 PUSH 指令之前　值 A　　值 B　　　值 C　　值 D
执行 PUSH 指令之后　值 A　　值 A　　　值 B　　值 C

（6）ENT　进入 ACCU 堆栈和 LEAVE　离开 ACCU 堆栈

指令格式为：ENT　LEAVE

ENT（进入累加器堆栈）将把 ACCU 3 的内容复制到 ACCU 4，并将 ACCU 2 的内容复制到 ACCU 3。如果在装载指令前面直接编程 ENT 指令，则可将中间结果保存到 ACCU 3 中。

LEAVE（离开累加器堆栈）将 ACCU 3 的内容复制到 ACCU 2，并将 ACCU 4 的内容复制到 ACCU 3。如果在移位或循环指令前面直接编程 LEAVE 指令，并将各累加器组合，则 LEAVE 指令就可起到数学运算指令的作用。ACCU 1 的内容和 ACCU 4 的内容保持不变。

例：

```
L    DBD0          //从数据双字 DBD0 中将值载入 ACCU 1（该值必须以浮点格式
                     表示）
L    DBD4          //将值从 ACCU 1 复制到 ACCU2。从数据双字 DBD4 中将值载
```

```
                    入 ACCU 1（该值必须以浮点格式表示）
+R                  //将 ACCU 1 和 ACCU 2 的内容作为浮点数（32 位）相加，结
                      果存到 ACCU 1
L    DBD8           //将值从 ACCU 1 复制到 ACCU 2，并从数据双字 DBD8 中将值
                      送入 ACCU 1
ENT                 //将 ACCU 3 的内容复制到 ACCU 4。将 ACCU 2 的内容（中间
                      结果）复制到 ACCU 3
L    DBD12          //从数据双字 DBD12 中将值载入 ACCU 1
-R                  //从 ACCU 2 的内容中减去 ACCU 1 的内容，并将结果保存在
                      ACCU 1 中
LEAVE               //将 ACCU 3 的内容复制到 ACCU 2。将 ACCU 4 的内容复制到
                      ACCU 3
/R                  //将 ACCU 2（DBD0 + DBD4）的内容除以 ACCU 1（DBD8 -
                      DBD12）的内容，结果存入 ACCU 1 中
T    DBD16          //将结果（ACCU 1）传送到数据双字 DBD16
```

（7）INC　增加 ACCU 1-L-L 和 DEC　减少 ACCU 1-L-L

指令格式为：INC <8 位整数>　　DEC <8 位整数>

ACCU 1-L-L 加常数（减常数），8 位整数常数的范围为 0～255。

INC <8 位整数>（增加 ACCU 1-L-L）将 ACCU 1-L-L 的内容加上 8 位整数，DEC <8 位整数>（减少 ACCU 1-L-L）从 ACCU 1-L-L 的内容中减去 8 位整数，并将结果均存储在 ACCU 1-L-L 中。ACCU 1-L-H、ACCU 1-H 和 ACCU 2 保持不变。执行该指令时不考虑状态位，也不会影响状态位。

该指令不适合 16 位或 32 位数学运算，因为从累加器 1 的低字的低字节到累加器 1 的低字的高字节不会产生进位。对于 16 位或 32 位数学运算，要使用 +I 或 +D 指令。

例：

```
L    MB22           //载入 MB22 的值
INC  1              //指令"ACCU 1（MB22）加 1"；结果存储在 ACCU 1-L-L 中
T  MB22             //将 ACCU 1-L-L 的内容（结果）传送回 MB22
L  MB250            //载入 MB250 的值
DEC  1              //指令"ACCU 1-L-L 减 1"，结果存储在 ACCU 1-L-L 中
T  MB250            //将 ACCU 1-L-L 的内容（结果）传送回 MB250
```

（8）+AR1　将 ACCU 1 加到地址寄存器 1 和 +AR2　将 ACCU 1 加到地址寄存器 2

指令格式为：　+AR1　　　+AR2

+AR1　<P#Byte. Bit>　　　+　<P#Byte. Bit>

参数 <P#Byte. Bit>（被加到 AR1 或 AR2 上的地址）的数据类型为指针常数。

+AR1 或 +AR2（加到 AR1 或加到 AR2）将在语句或 ACCU 1-L 中指定的偏移量加到 AR1（AR2）的内容上。首先将整数（16 位）扩展为符号正确的 24 位，然后将其加到 AR1 的最低有效的 24 位（AR1 中的相对地址的一部分）。在 AR1（AR2）中，区域 ID 的部分（位 24、25 和 26）保持不变。执行该指令时不考虑状态位，也不会影响状态位。

+AR1、+AR2：要加到 AR1、AR2 的内容中的整数（16 位）由 ACCU 1-L 中的值指定。允许的值范围为 -32768～+32767。

+AR1 <P#Byte. Bit>、+AR2 <P#Byte. Bit>：要加的偏移量由 <P#Byte. Bit> 地址指定。

例 1：

L　+300　　　　　　//将值载入 ACCU 1-L

+AR1　　　　　　　//将 ACCU 1-L（整数，16 位）加到 AR1

例 2 ：

+AR1　P#300. 0　　　//将偏移量 300. 0 加到 AR1

例 3：

L　+300　　　　　　//载入 ACCU 1-L 中的值

+AR1　　　　　　　//将 ACCU 1-L（整数，16 位）加到 AR2

例 4 ：

+AR1　P#300. 0　　　//将偏移量 30. 0 加到 AR2

（9）BLD　　程序显示指令（空）

指令格式为：BLD <数字>

<数字>　指定 BLD 指令，范围为 0～255。

BLD <数字>（程序显示指令；空指令）不执行任何功能，并且不影响状态位。该指令用于编程设备（PG）的图形显示。在 STL 中显示梯形图或 FBD 程序时将自动创建它。地址 <数字> 指定 BLD 指令并由编程设备生成。

（10）NOP 0　　空指令和 NOP 1　　空指令

指令格式为：NOP 0　　　NOP 1

NOP 0（地址为“0”的指令 NOP）不执行任何功能，并且不影响状态位。该指令代码包含具有 16 个零的位模式。当显示某个程序时，该指令仅对编程设备（PG）有用。

NOP 1（地址为“1”的指令 NOP）不执行任何功能，并且不影响状态位。该指令代码包含具有 16 个“1”的位模式。当显示某个程序时，该指令仅对编程设备（PG）有影响。

第五章　西门子编程软件STEP 7

本章主要掌握硬件组态和参数设置各个环节的含义及配置过程。掌握定义符号、生成块、下载与上传、参考数据显示以及程序调试等的方法和步骤。了解故障诊断的方法和过程。

第一节　STEP 7 编程软件的使用简介

S7 -300/400 系统的组态与编程是在 STEP 7 软件上进行的，本节以 S7 -300 为例简要介绍利用 STEP 7 进行编程的基本概念、方法和步骤。

一、STEP 7 概述

STEP 7 编程软件用于对 SIMATIC S7 -300/400、SIMATIC M7-300/400、SIMATIC C7 等系统进行编程和开发。

STEP 7 中是通过项目的方式来管理自动化系统，其功能包括硬件组态（配置）、参数设置、网络组态、通信连接、创建符号、编程、组态消息和操作员监控变量、起动和运行维护、监视、诊断、文档创建和归档等。

二、STEP 7 标准软件包

STEP 7 软件包符合面向图形和对象的 Windows 操作原则，可运行在在 Windows 2000、Windows XP、Windows Server 2003 下，为适应不同的应用对象，可选择不同的版本。STEP 7 标准软件包的组成如图 5-1 所示。

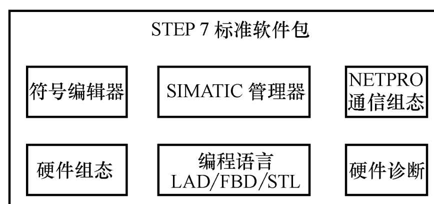


图 5-1　STEP 7 标准软件包的组成

1）SIMATIC 管理器可浏览 SIMATIC S7 、M7、C7 的所有工具软件和数据。

2）符号编辑器管理所有的全局变量，用于定义符号名称、数据类型和全局变量的注释。

3）通信组态包括组态的连接和显示、定义 MPI 或 PROFIBUS-DP 设备之间由时间或事件驱动的数据传输、定义事件驱动的数据、用编程语言对所选通信块进行参数设置。

4）硬件组态用于对硬件设备进行配置和参数设置。包括系统组态（选择机架、给各个槽位分配模块、自动生成 I/O 地址。）、CPU 参数设置（例如起动特性、扫描监视时间等）和模块参数设置（用于定义硬件模块的可调整参数）。

5）编程语言工具中可以使用梯形图语言（LAD）、功能块图语言（FBD）和语句表语言（STL）。

6）硬件诊断工具为用户提供自动化系统的状态，可快速浏览 CPU 的数据以及用户程序运行中的故障原因，也可用图形方式显示硬件配置，例如模块的一般信息和状态、显示模块故障、显示诊断缓冲区信息等。

三、STEP 7 的授权

使用 STEP 7 编程软件，需要一个产品专用的许可证密钥（用户权限）。从 STEP 7 V5. 3 版本

起，该密钥（授权）通过自动化许可证管理器安装。

自动化许可证管理器是 Siemens AG 的软件产品，它用于管理所有系统的许可证密钥（许可证模块）。

自动化许可证管理器集成了自身的在线帮助。要在安装许可证管理器后获取帮助，请按“F1”键或选择帮助→许可证管理器帮助菜单命令。该在线帮助包含自动化许可证管理器功能和操作的详细信息。

没有授权也可以使用 STEP 7，以便熟悉用户接口和功能，但是在使用时每隔一段时间将会搜索授权，提醒使用者安装授权，只有安装了授权才能有效地使 STEP 7 工作。如果因为硬盘出现故障而丢失授权，可以使用试用许可证，它允许 STEP 7 继续运行一段有限的时间。在此期间应与当地西门子代表处联系，以获得丢失授权替换授权 Author。

合法使用受许可证保护的 STEP 7 程序软件包时必须要有许可证。许可证为用户提供使用产品的合法权限。许可证证书（CoL）和许可证密钥提供使用权限证明。

SIEMENS AG 给受许可证保护的所有软件颁发许可证密钥。起动计算机后，只能在确认具有有效许可证密钥之后，才能根据许可证和使用条款使用该软件。

自动化许可证管理器通过 MSI 设置过程安装。STEP 7 产品 CD 包含自动化许可证管理器的安装软件。可以在安装 STEP 7 的同时安装自动化许可证管理器或在以后安装。随后安装许可证密钥，起动 STEP 7 软件时如果没有可用的许可证密钥，将显示一个指示该情况的警告消息。

可从软盘上安装许可证密钥，或安装从 Internet 上下载的许可证密钥（这种情况下，必须首先订购许可证密钥和使用网络中可用的浮动许可证密钥的方式安装许可证密钥）。

可以使用不带许可证密钥的标准软件来熟悉用户接口和功能，不能无限制地使用 STEP 7 软件。

四、STEP 7 的安装和硬件接口

STEP 7 安装程序可自动完成安装。通过菜单可控制整个安装过程，可通过标准 Windows 2000/XP/Server 2003 软件安装程序执行安装。

1. 安装的主要步骤

1）将数据复制到编程设备中。

2）组态 EPROM 和通信驱动程序。

3）安装许可证密钥（如果需要）。

2. 安装要求

1）操作系统：Microsoft Windows 2000 或 Windows XP、Windows Server 2003。

2）基本硬件：包含下列各项的编程设备或 PC：奔腾处理器（600 MHz）、至少 256MB 的 RAM、彩色监视器、键盘和鼠标，Microsoft Windows 支持所有这些组件。

3. 编程设备（PG）

编程设备是具有特殊紧凑型设计、用于工业用途的 PC。它配备齐全，可用来对 SIMATIC PLC 进行编程。

1）硬盘空间：请参见“README. WRI”文件，获取所需硬盘空间信息。

2）MPI 接口（可选）：只有在 STEP 7 下通过 MPI 与 PLC 通信时才要求使用 MPI 接口来互连 PG/PC 和 PLC。此时需要：一个与设备通信端口连接的 PC USB 适配器，或者在设备中安装 MPI 模块（例如，CP5611）。

PG 装配有 MPI 接口。只有在通过 PC 编程 EPROM 时，才要求使用外部存储器（可选）。

PC/MPI 适配器 + RS-232C 通信电缆用于接口连接。PC 的通信卡 CP 5611（PCI 卡）、CP

5511 或 CP 5512（PCMCIA 卡）将 PC 连接到 MPI 或 PROFIBUS 网络。计算机的工业以太网通信卡 CP 1512（PCMCIA 卡）或 CP 1612（PCI 卡），通过工业以太网实现计算机与 PLC 的通信。

五、STEP 7 的编程功能

1. 编程语言

STEP 7 标准软件包配备三种基本编程语言，即梯形图（LAD）、功能块图（FBD）和语句表（STL）编程语言，在 STEP 7 中可以相互转换。

梯形图（LAD）是 STEP 7 编程语言的图形方式，与继电器控制电路图相似，可跟踪能流的流动，是一种融逻辑操作、控制于一体，面向对象的、实时的、图形化的编程语言。

语句表（STL）是 STEP 7 编程语言的文本方式，用助记符来表达 PLC 的各种控制功能。为便于编程，语句表做了扩展，可调用一些高级语言结构（如结构化数据访问和块参数）。

功能块图（FBD）是 STEP 7 编程语言的另一种图形表示，类似于普通逻辑功能图，它沿用了半导体逻辑电路的逻辑框图的表达方式。用逻辑框表示逻辑功能。复杂功能（如算术功能）可直接结合逻辑框表示，功能块图通过软连接的方式把所需的功能块图连接起来，用于实现系统的控制。功能块图（FBD）的表达格式有利于程序流的跟踪。

选件包提供其他编程语言，包括 S7 Graph、S7 SCL、S7-HiGraph 和 S7 CFC。

S7 Graph 用于编制顺序控制程序，特别适合于生产制造过程。

S7 SCL 是用于实现复杂的数学运算的高级文本语言，适合于熟悉高级编程语言的用户使用，进行计算和数据处理。

S7 HiGraph 是用状态图（State Graphs）描述异步、非顺序过程的编程语言。

S7 CFC 是通过把程序库中以块形式提供的各种功能用图形连接实现编程的语言，适合于连续过程控制的编程。

2. 符号编辑器

符号编辑器可管理所有共享符号，通过符号编辑器创建的符号表可供所有其他工具使用。符号编辑器具有以下几个方面的功能。

1）设置过程信号（输入/输出）、位存储器以及块的名称和注释。

2）Windows 程序间的互相导入/导出。

3）分类排序。

3. 增强的测试和服务功能

测试和服务功能包括设置断点、强制输入和输出、多 CPU 运行（仅限于 S7-400）、重新布线、显示交叉参考表、状态功能、直接下载和调试块、同时监测几个块的状态等。

程序中的特殊点可以通过输入符号名或地址快速查找。

4. 在线帮助

使用在线帮助可迅速、快捷地访问各种信息而无需再查询各种手册。通过帮助菜单，也可从每个不同窗口访问与当前对话状况相关的各个主题。

选择菜单栏帮助菜单中的菜单命令，单击对话框中的“帮助”按钮。随后将显示关于该对话框的帮助，将光标置于窗口或对话框中需要获得帮助的主题上，然后按“F1”键或选择菜单命令帮助→上下文关联帮助或使用窗口中的问号符号光标来调用在线帮助。

六、STEP 7 的硬件组态与诊断功能

1. 硬件组态（配置）

1）系统组态：选择硬件机架，模块分配给机架中希望的插槽。

2）CPU 的参数设置。

3）模块的参数设置。可以防止输入错误的数据。

2. 通信组态（配置）

1）网络连接的组态和显示。

2）设置用 MPI 或 PROFIBUS-DP 连接的设备之间的周期性数据传送的参数。

3）设置用 MPI、PROFIBUS 或工业以太网实现的事件驱动的数据传输，用通信块编程。

3. 系统诊断

1）快速浏览 CPU 的数据和用户程序在运行中的故障原因。

2）用图形方式显示硬件配置、模块故障；显示诊断缓冲区的信息等。

第二节 硬件组态与参数设置

一、项目的创建与项目的结构

项目用于存储在提出自动化解决方案时所创建的数据和程序。项目所汇集的数据包括：关于模块硬件结构及模块参数的组态数据、用于网络通信的组态数据和用于可编程模块的程序。在创建项目时的主要任务就是准备这些数据，以备编程使用。

1. 项目的创建

创建项目时，可使用向导创建项目。首先双击 SIMATIC 管理器图标进入 SIMATIC Manager（SIMATIC 管理器）窗口，弹出新项目小窗口，点击“下一个”选择 CPU 模块型号、需要生成的逻辑块和输入项目名称，如图 5-2 所示。

点击“完成”生成的项目如图 5-3 所示。生成项目后，可以先组态硬件，然后生成软件程序，也可以先生成软件后组态硬件。

也可手动创建项目，在 SIMATIC 管理器中使用菜单命令文件→新建（菜单命令均用加粗文字表述，以下均同）来创建一个新项目。它已经包含“MPI 子网”对象。

2. 项目的结构和窗口

数据将以对象的形式存储在项目中。对象在项目中按树形结构排列（项目体系）。项目体系在项目窗口中的显示类似于 Windows 资源管理器中的显示。只是对象图标的外观不同。项目的结构分三层，第一层为项目，第二层为子网、站或 S7 /M7 程序，第三层取决于第二层的对象。

项目窗口分为两半部分：左半部分表示项目的树形结构；右半部分表示所选视图左半部分已打开的对象所包含的对象（大图标、小图标、列表或详细信息）。

对象体系的最上端是代表整个项目的对象“电机起动”的图标。它可用于显示项目属性，并可用作网络文件夹（用于对网络进行组态）、站文件夹（用于对硬件进行组态）以及 S7 或 M7 程序的文件夹（用于创建软件）。项目中的对象在选择项目图标时均将显示在项目窗口的右半部分。该类型对象体系最上端的对象（库以及项目）构成了用于对对象进行选择的对话框的起始点。

项目对象中包含站对象和 MPI 对象。站对象包含硬件和 CPU，SIMATIC 300/400 站表示具有一个或多个可编程模块的 S7 硬件配置。CPU 包含 S7 程序和连接。S7 程序包含了用于 S7 /M7CPU 模块的软件或用于非 CPU 模块（例如，可编程 CP 或 FM 模块）的软件（源文件、块和符号表）。源文件文件夹包含了文本格式的源程序。离线视图的块文件夹可包括：逻辑块（OB、FB、FC、SFB、SFC）、数据块（DB）、自定义的数据类型（UDT）和变量表。在线视图的块文件夹包括已经下载给 PLC 的可执行程序部分。系统数据对象表示系统数据块生成程序时会自动生成一个空的符号表。

为在项目中创建一个新站，可打开项目，以便显示项目窗口。其方法是先选择项目再通过使

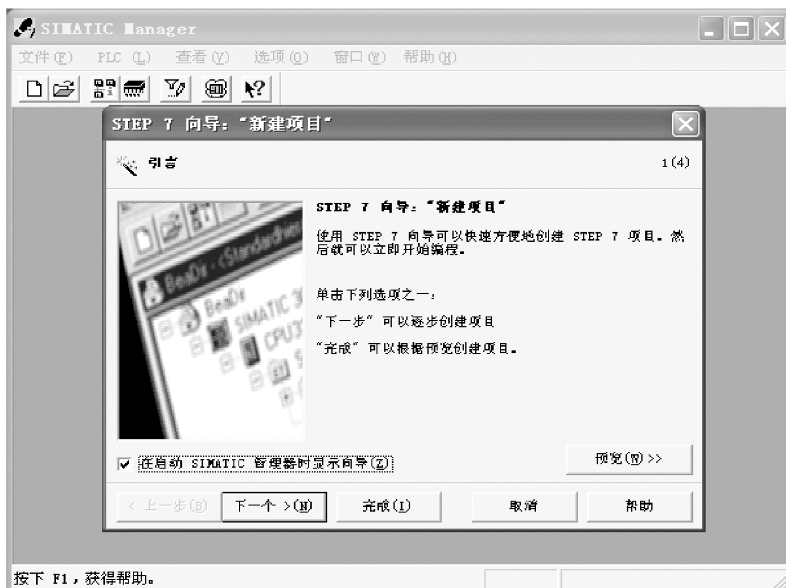
用菜单命令插入→站点，为需要的硬件创建“站”对象。如果站没有显示，单击项目窗口中项目图标前的“+”号，如图 5-4 所示。也可以用右键功能创建。

二、硬件组态

1. 硬件组态介绍

硬件组态指的是在站窗口中对机架、模块、分布式 I/O（DP）机架以及接口子模块等进行排列。像实际的机架一样，可在其中插入特定数目的模块。

在组态表中，STEP 7 自动给每个模块分配一个地址。如果站中的 CPU 可自由寻址（可为模块的每个通道自由分配一个地址，而与其插槽无关），可改变站中模块的地址。



a)



b)

图 5-2 项目创建



c)



d)

图 5-2　项目创建（续）



图 5-3　生成的项目



图 5-4　插入站点

在 PLC 起动时，CPU 将比较 STEP 7 中创建的预置组态与设备的实际组态。如果两者不符，将立即产生错误报告。S7 PLC 和模块的属性均可预先设置为默认值，在大多情况下，都不需要对其进行组态。

组态时设置的 CPU 的参数保存在系统数据块 SDB 中，其他模块的参数保存在 CPU 中。在更换 CPU 之外的模块后不需要重新对它们赋值，其原因是在 PLC 起动时 CPU 自动地向其他模块传送设置的参数。对于网络系统，需要对以太网、PROFIBUSI - DP 和 MPI 等网络的结构和通信参数进行组态，将分布式 I/O 连接到主站。例如可以将 MPI（多点接口）通信组态为时间驱动的循环数据传送或事件驱动的数据传送。对于硬件已经装配好的系统，用 STEP 7 建立网络中各个站对象后，可以通过通信从 CPU 中读出实际的组态和参数。

2. 硬件组态的步骤

组态 PLC 会用到两个窗口，为站结构放置机架的站窗口和“硬件目录”窗口，可以从“硬件目录”窗口中选择所需要的硬件组件，如机架、模块以及接口子模块。如果没有出现“硬件目录”窗口，可选择菜单命令“选项→目录”。该命令可打开、关闭硬件目录的显示。硬件组态窗口如图 5-5 所示。

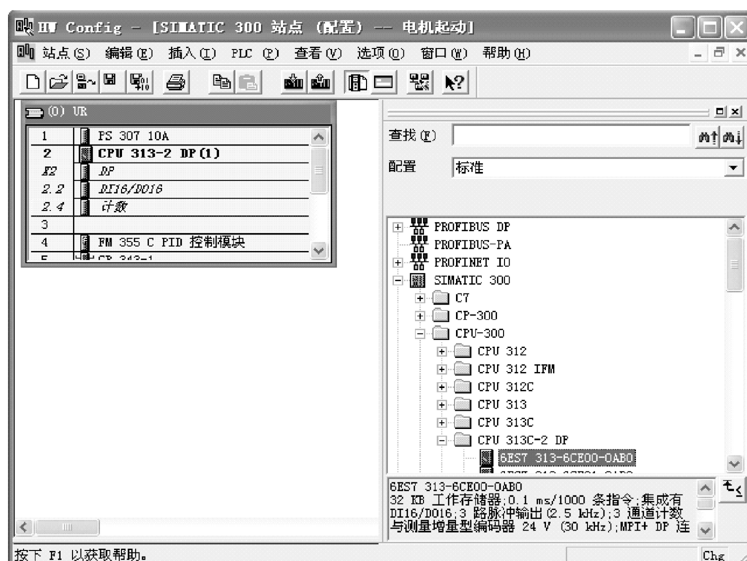


图 5-5　硬件组态窗口

按照下列步骤进行组态：

1）在“硬件目录”中选择一个硬件组件。

2）通过拖放操作或双击模块，将选定组件复制到站窗口中。

3）保存硬件设置，并将它下载到 PLC 中去。

3. 设置组件属性

一旦在站窗口中排列了组件，通过双击该组件，或选择菜单命令“编辑→对象属性”；或者将光标移到组件上，按下鼠标右键，然后从弹出菜单中选择对象属性命令来改变默认属性的对话框（参数或地址）。

三、CPU 模块的参数设置

在硬件配置的站窗口中，双击 CPU 所在的行，弹出属性窗口，即可进行 CPU 的参数设置。CPU 属性设置对话框如图 5-6 所示。

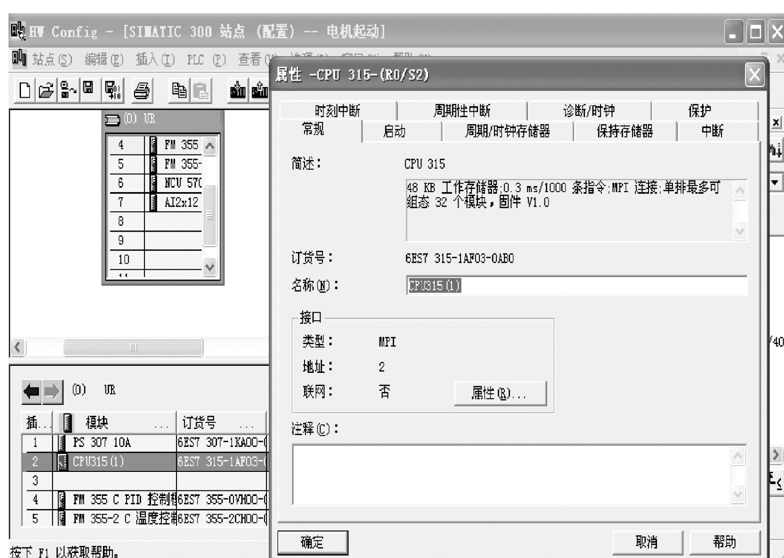


图 5-6　CPU 属性设置对话框

四、数字量输入模块的参数设置

I/O 模块的参数设置均在 CPU 处于 STOP 模式下进行。设置完后下载到 CPU 中。当 CPU 从 STOP 模式转换为 RUN 模式时，CPU 将参数传送到每个模块。在硬件配置的站窗口中，双击模块所在的行，弹出属性窗口，即可进行相应模块的参数设置，如图 5-7 所示。



图 5-7　数字量输入模块的参数设置

五、数字量输出模块的参数设置

在硬件配置的站窗口中，双击模块所在的行，弹出属性窗口，即可进行相应模块的参数设置，如图 5-8 所示。

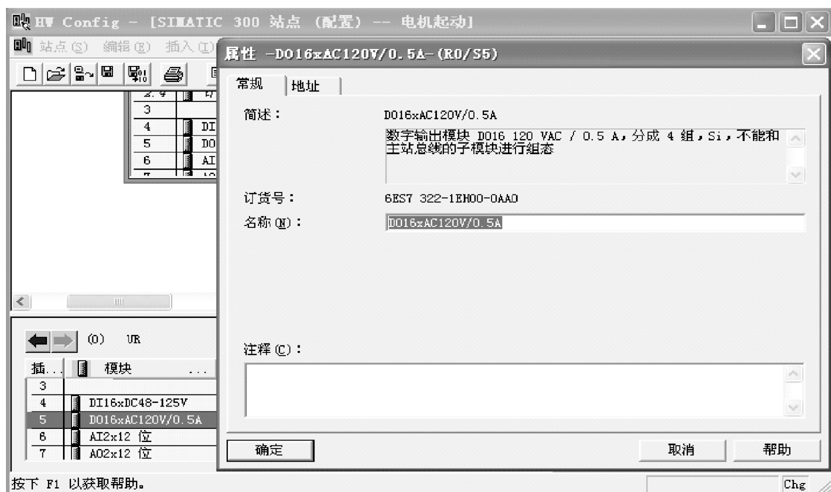


图 5-8 数字量输出模块的参数设置

六、模拟量输入模块的参数设置

在硬件配置的站窗口中，双击模块所在的行，弹出属性窗口，即可进行相应模块的参数设置，如图 5-9 所示。

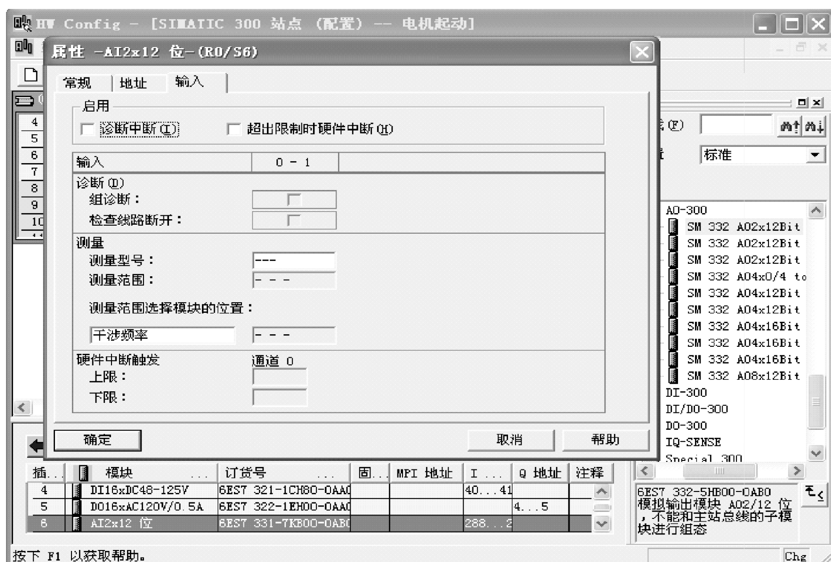


图 5-9 模拟量输入模块的参数设置

七、模拟量输出模块的参数设置

在硬件配置的站窗口中，双击模块所在的行，弹出属性窗口，即可进行相应模块的参数设置，如图 5-10 所示。



图 5-10　模拟量输出模块的参数设置

第三节　定 义 符 号

在 STEP 7 程序中，使用地址如 I/O 信号、位内存、计数器、定时器、数据块和功能块。可以在程序中访问这些地址，但是如果使用地址符号，程序将更容易阅读。然后，可以通过此符号访问用户程序中的地址。在编程语言梯形图、功能块图和语句表中，可以输入地址、参数和块名称，作为绝对地址或符号。使用菜单命令“查看→显示→符号表示法”，可以在地址的绝对表示法和符号表示法之间切换。

绝对地址包含地址标识符和内存位置（例如，Q 4.0，I 1.1，M 2.0，FB21）。如果将符号名分配给绝对地址，可以使程序更易读，并能简化故障排除。STEP 7 可以自动地将符号名称翻译成所需要的绝对地址。如果要使用符号名称访问 ARRAY、STRUCT、数据块、本地数据、逻辑块和用户自定义数据类型，在使用符号寻址数据前，必须首先将符号名称分配给绝对地址。使用符号地址，更容易识别程序中的元素与过程控制项目的组件的匹配程度。

为了更容易地使用符号地址编程，可以显示绝对地址和属于符号的符号注释。可以使用菜单命令“查看→显示→符号信息”激活此信息。这意味着每个 STL 语句后的行注释中包含更多的信息。不能编辑该显示；任何改变都必须在符号表或变量声明表中进行。图 5-11 显示了在 STL 中的符号信息。

```
程序段?1: AND 扫描
当 "Key 1" (I 0.1)和 "Key 2" (I 0.2)同时激活时(即，其信号状态为 "1" = 24V),
"Green Light" (Q 4.0)也激活。

为了点亮交通灯，必须同时激活两个输入。

      A     "Key_1"            I0.1        -- 对于 AND 查询
      A     "Key_2"            I0.2        -- 对于 AND 查询
      =     "Green_Light"      Q4.0        -- AND 查询结果
```

图 5-11　查看符号信息显示

共享符号与局域符号

符号寻址允许用户用有一定含义的符号地址来代替绝对地址。将短的符号和长的注释结合起来使用，可使程序更简单。

1. 共享符号

共享符号可以被所有的块使用，在所有的块中的含义是一样的。在整个用户程序中，同一个共享符号在整个用户程序中必须是唯一的。共享符号由字母、数字及特殊字符组成，可以用汉字来表示共享符号。

2. 局域符号

局域符号仅在对其进行定义的块中有效。同一个符号可以根据不同用途在不同的块中使用。局域符号只能使用字母、数字和下划线，不能使用汉字。

3. 显示共享符号与局域符号

来自符号表中的符号（共享符号）将显示在符号“..”内，来自块的变量声明表中的符号（局部符号）将在前面冠以字符“#”。“..”或“#”无须输入。在 LAD、FBD 或 STL 中输入程序时，语法检查将自动添加这些字符。如果担心在某些情况下出现混淆，例如在符号表和变量声明中都使用同一个符号，那么要使用该共享符号时，必须直接对其进行编码（输入地址或者包括引号的符号）。此时，没有进行分别编码的任何符号都将解释为指定块（局部）的变量。如果符号包含有空格，也必须对共享符号进行编码（输入地址或者包括引号的符号）。当在 STL 源文件中进行编程时，将采用同样的特殊字符及准则。在自由编辑模式下，将不会自动添加代码字符，但如果希望避免混淆，这些代码字符将仍然需要。

4. 设置地址优先级

在改变符号表中的符号、改变数据块或功能块的参数名称、改变引用组件名称的 UDT 或修改多重实例时，地址优先级有助于按意愿调整程序代码。为了设置地址优先级，进入 SIMATIC 管理器，并选择块文件夹，然后选择菜单命令“编辑→对象属性”。在“地址优先级”标签中，就可以进行与要求相适合的设置。设置地址优先级如图 5-12 所示。

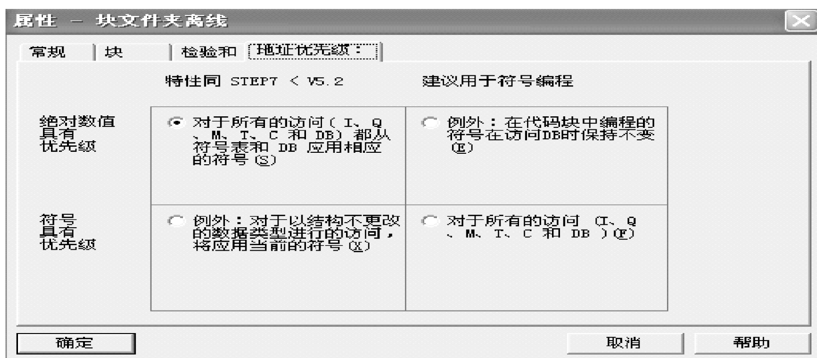


图 5-12　设置地址优先级

在 SIMATIC 管理器中，选择块文件夹，然后选择菜单命令“编辑→ 检查块一致性”。“检查块一致性”功能在单个块中进行必要的改动。此操作进行跟踪改动。

5. 符号表编辑

在创建 S7 或 M7 程序时，将自动创建一个（空的）符号表（“符号”对象）。在符号表中不能定义数据块中的地址（DBD、DBW、DBB 和 DBX），而应在数据块的声明表中定义。

创建符号表的过程是双击项目窗口中的 S7 程序或 M7 程序，对象“符号”显示在窗口的右半部分，如果符号表已删除或被覆盖，使用菜单命令“插入→符号表”以插入一个新的符号表，打开对象“符号”，可以通过双击此对象，显示所要编辑的符号表如图 5-13 所示。

在符号编辑器中可以进行符号编辑或查看，如图 5-14 所示。用菜单命令查看→列 R，O，M，



图 5-13　符号表

C，CC 可以选择是否显示表中的“R，O，M，C，CC”列，它们分别表示监视属性、在 WinCC 里是否被控制和监视、信息属性、通信属性和触点控制。可以用菜单命令“查看→排序选择符号”表中变量的排序方法。

输入符号的方法有三种。第一种是直接在符号表中输入符号及其绝对地址。如果希望输入许多符号，或者当您为了使已分配的符号在屏幕上显示而创建项目的符号表时，建议使用该方法，它能很容易地对符号进行总览。

第二种是通过对话框，在正在输入程序的窗口中打开一个对话框，然后定义一个新的符号或重新定义现有的符号。一般在定义单个的符号时使用该过程。

第三种是从其他表格编辑器中导入符号表，可在任何表格编辑器（例如 Microsoft Excel）中创建符号表的数据，然后将所创建的文件导入符号表。

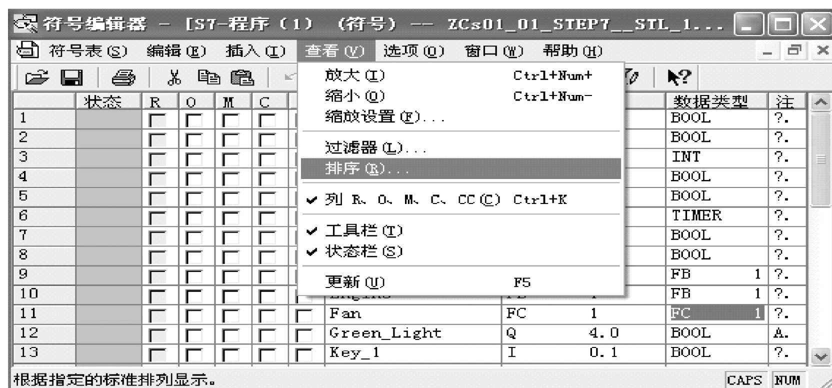


图 5-14　符号编辑器

第四节　创建逻辑块

一、块文件

创建 S7 CPU 程序包括块和源文件。使用 S7 程序下的文件夹“块”来存储块，块文件如图 5-15 所示。该块文件包含有完成自动化任务而需要下载给 S7 CPU 的块。这些可装载的块包括逻

辑块（OB、FB、FC）、数据块（DB）。在块文件中将自动创建一个空的组织块 OB1，因为在执行 S7 CPU 中的程序时将始终需要这个块。

块文件还包含创建的用户自定义数据类型（UDT）（这些类型将使编程更容易，且不需要将其下载给 CPU）、为在调试程序时对变量进行监视和修改而创建的变量表（VAT）。不需要将变量表下载给 CPU。包含有系统信息（系统组态、系统参数等）的对象“系统数据”（系统数据块）。在组态硬件时将创建并提供这些系统数据块。在用户程序中需要调用系统功能（SFC）与系统功能块（SFB）。但自己不能编辑 SFC 与 SFB。

除了系统数据块（只能通过 PLC 的组态对其进行创建和编辑）外，用户程序中的块都要使用各自的编辑器进行编辑。对应的编辑器要通过双击相应块起动。



图 5-15 块文件

二、逻辑块的创建

逻辑块包括 OB、FB、FC。

1. 程序的输入方式

程序的输入方式有增量输入方式（每一行和每个元素输入均无错误才能完成当前输入）或源代码方式（或称文本方式、自由编辑方式，可快速输入程序）。

2. 生成逻辑块

使用 SIMATIC 管理器创建块，如选择菜单命令“插入→S7 块→功能块（FB）”，如图 5-16 所示。



图 5-16 创建逻辑块

三、程序编辑器窗口的结构

程序编辑器的窗口可拆分为以下几个部分。

1. 总览

“程序元素”标签将显示一个程序元素表格，其中的程序元素均可插入到 LAD、FBD 或 STL 程序中。“调用结构”标签表示当前 S7 程序中的块的调用层次。

2. 变量声明

变量声明分为“变量表”和“变量详细视图”部分。见图 5-17 右上部分。

3. 指令

指令表显示了将由 PLC 进行处理的块代码。它由一个或多个程序段组成。见图 5-17 右中间部分。

4. 详细资料

“详细资料”窗口（见图 5-17 下方）中的各种不同标签提供了众多的功能，例如，用于显示出错消息、对符号进行编辑、生成地址信息、对地址进行控制、对块进行比较的功能以及对硬件诊断时的出错定义进行编辑的功能。



图 5-17　程序编辑器

四、程序指令输入

1. 梯形图指令输入

选择菜单命令“查看→LAD”进入梯形图编程界面。选择菜单命令“插入→程序段”添加程序段，然后在程序段中选择将在其后插入梯形图元素的点。其次通过以下方法之一，在程序段中插入所需要的元素。梯形图指令输入如图 5-18 所示。

1）选择菜单命令“插入→程序元素”以便打开“程序元素”标签，并在目录中选择所需要的元素。

2）单击工具栏中的常开触点、常闭触点或输出线圈的按钮。

3）在“插入”菜单中选择相应的菜单命令，例如，“插入→LAD 语言元素→常开触点”。

通过选择现有的梯形元素，然后从“编辑→剪切”、“编辑→复制”或“编辑→粘贴”中选择一个菜单命令，也可编辑代码段。

2. 梯形图指令输入规则

一个梯形图程序段可由多个分支中的许多元素组成。所有的元素和分支必须进行连接；左电源线不算作连接（IEC 1131-3）。每个梯形图程序段都必须使用线圈或逻辑方框来关闭。不能使用比较框，中间变量输出（_⁄（#）_⁄）和用于上升沿（_⁄（P）_⁄）或下降沿



图 5-18 梯形图指令输入

(_/（N）_/)计算的线圈指令放置于分支开始的最左边或结束程序段。

3. 语句表指令输入

逻辑块的代码段通常包含许多程序段，这些程序段则由语句列表组成。在代码段中，可编辑块标题（最多 64 个字符）、块注释（对整个逻辑块进行记录，例如，块的用途）、程序段标题（最多 64 个字符）、程序段注释（记录单个程序段的功能）以及程序段内的语句行。选择菜单命令“查看→STL”进入语句表编程界面。

语句由标记（可选）、指令、地址和注释（可选）组成。在逻辑块的代码段中，可输入块标题和程序段标题，以及块注释或程序段注释。每条语句均单独占一行。在一个块中，最多可输入 999 个程序段。可将光标放置在块名称或程序段名称右边的单词“标题”上（例如，程序段 1：标题：）单击，即可打开一个可在其中输入标题的文本框，语句表指令输入如图 5-19 所示。



图 5-19 语句表指令输入

在语句表编程语言表达式中，可为每条语句输入一条注释，其方法是在每个地址或符号名称后，按下空格键，使用双斜杠（//）作为语句注释的开头，通过按下回车键完成注释的输入。

4. 打开和编辑块的属性

选择菜单命令“文件→程序段”来查看和编辑块属性，如图 5-20 所示。

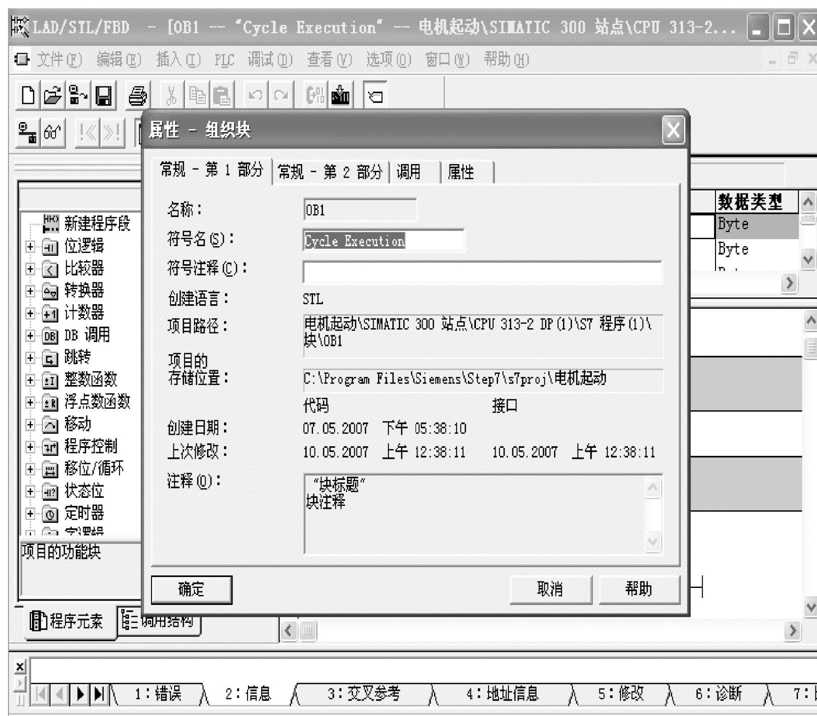


图 5-20　块属性

五、程序下载和上传

1. 下载准备

编程设备和 PLC 中的 CPU 之间存在一个可用连接（例如，通过多点接口）用于访问 PLC。为将块下载给 PLC，在项目的对象属性对话框中的“使用”选择条目“STEP 7”（见图 5-21）。要下载的程序已完成编译无错误。CPU 必须处于允许进行下载的工作模式（STOP 或 RUN-P）。

注意，在 RUN-P 模式下，程序每次下载一个块，如果通过这样来覆盖旧的 CPU 程序，则可能会导致冲突。建议在下载之前将 CPU 切换到 STOP 模式。在下载用户程序之前，应复位 CPU，以确保 CPU 上没有任何“旧的”块。

2. 下载方法

完成组态、参数分配和程序创建并建立在线连接后，就可将完整的用户程序或各个块下载至 PLC。使用下载功能将用户程序或可装入对象（例如块）下载到 PLC。如果块已存在于 CPU 的 RAM 中，将提示您确认是否要覆盖块。

在项目窗口中选择可加载的对象，通过 SIMATIC 管理器的菜单命令“PLC→下载”进行下载，窗口如图 5-22 所示。

对于编写块、组态硬件和网络的下载，可通过正在使用的应用程序主窗口中的菜单（菜单命令：PLC→下载）直接下载当前正在编辑的项目。

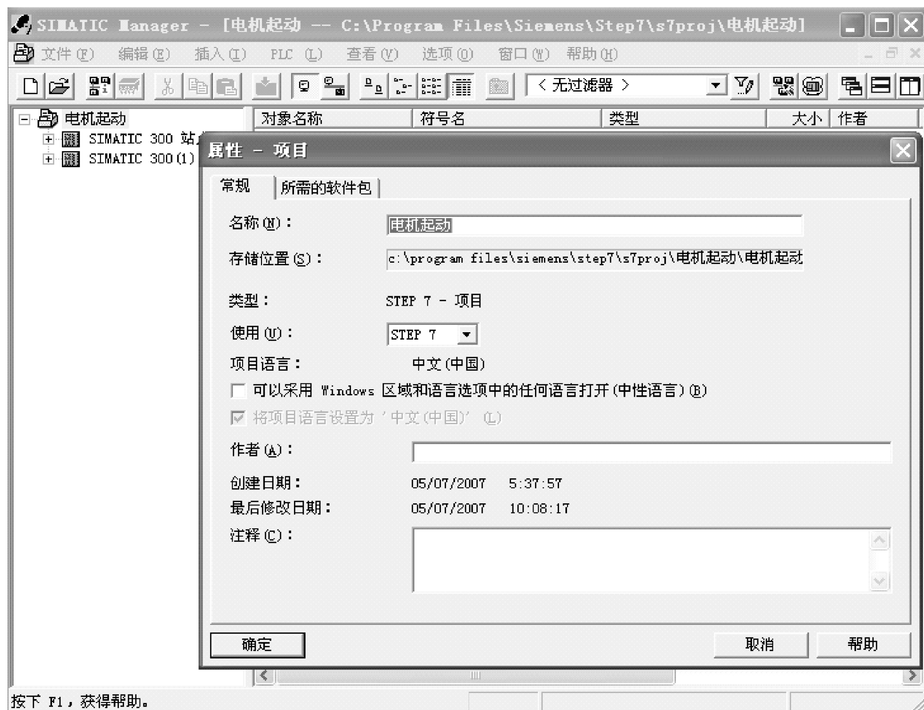


图 5-21 项目属性设置



图 5-22 PLC 下载设置窗口

也可以打开具有 PLC 视图的在线窗口（例如，使用“视图→在线”或“PLC→显示”可访问节点），将想要下载的对象复制到在线窗口。

3. 程序上载

通过 SIMATIC 管理器的菜单命令“PLC→上传”（窗口见图 5-22）将 PLC 当前内容下载到 PG/PC 上的编程软件打开的项目中，此操作将该项目原来的内容覆盖。

使用菜单命令“PLC→将站点上传到 PG”，可以将当前组态和所有块从所选的 PLC 上传到编程设备。

第五节　仿真软件使用与说明

使用 PLCSIM 5.4，可以仿真 S7PLC 的操作，可以测试控制程序，无需连接 S7 硬件。本节介绍的仿真软件面向那些具有 S7 PLC 和 STEP 7 编程知识与经验的开发人员、程序员以及维修人员，适用于 V5.4 SP 5 或更高版本的 S7-PLCSIM 仿真软件。

S7-PLCSIM V5.4 具有以下属性：

1）优化下载方案。

2）支持额外的 PG/PC 接口：PCinternal（本地）。

3）简化访问方式。

4）优化 WinCC 与 WinCC flexible 之间的通信。

5）在状态栏中显示所有 CPU 访问地址。

在 S7-PLCSIM 中，可以在仿真 PLC 中执行以及测试 STEP 7 用户程序。仿真在 PC 或编程设备（如 Field PG）中执行。由于仿真是完全在 STEP 7 软件中实施的，因此不需要任何 S7 硬件（CPU 或信号模块）。可以使用 S7-PLCSIM 仿真专为 S7-300、S7-400 以及 WinAC 控制器开发的 STEP 7 用户程序。S7-PLCSIM 提供一个简单的 STEP 7 用户程序界面，以供监视以及修改诸如输入和输出变量这样的不同对象。当在仿真 CPU 上运行程序时，还可以使用 STEP 7 软件的各个应用程序。例如，可使用诸如变量表（VAT）这样的工具来控制和监视变量。S7-PLCSIM 提供一个用于查看和修改控制程序变量、在单次或持续扫描模式中运行仿真 PLC 程序、更改仿真控制器的工作模式的图形用户界面。

此外，S7-PLCSIM 还包含名为 S7ProSim 的 COM 对象，此对象提供对仿真 PLC 的编程访问。通过 S7ProSim，可以编写软件以执行诸如更改仿真 PLC 的钥匙开关位置、在单次扫描模式中运行控制程序以及读取或写入控制器值这样的任务和许多其他任务。

S7-PLCSIM 具有如下功能：

1）起动时，打开现有仿真。

2）在仿真 PLC 上运行专为 S7-300、S7-400、T-CPU 以及 WinAC PLC 设计的程序。

3）创建视图对象，这可以访问仿真 PLC 的输入和输出存储区、累加器以及寄存器。

4）通过符号寻址访问存储器。

5）自动运行定时器。

6）手动置位定时器，或者复位所有定时器或一个定时器。

7）切换 CPU 工作模式（STOP、RUN 以及 RUN-P）。

8）使用“暂停”（Pause）菜单命令暂停仿真，而不影响程序的状态。

9）使用错误和中断 OB 测试程序的运行情况。

10）记录一系列事件（修改输入和输出存储区、位存储器、定时器以及计数器）。

11）回放程序记录，以便自动测试。

在 STEP7 中集成

可以对仿真 PLC 使用所有 STEP 7 工具。尽管仿真 PLC 完全存在于软件中，但 STEP 7 会将仿真 PLC 作为真正的 S7 PLC 来使用，几乎不存在任何区别。

一、与“真正”PLC 的区别

1. 仿真 PLC 的功能

仿真 PLC 提供以下“真正”PLC 所不具备的功能：

1）“暂停”命令暂停仿真 CPU，并允许从程序暂停的指令处继续执行程序。

2）如果在 STOP 模式中使用仿真 CPU，则 S7-PLCSIM 不会更改输出状态。如果选择 RUN 模式选择器位置，则将无法下载 STEP 7 用户程序或使用 STEP 7 工具更改任何参数。真正的 S7 PLC 允许在设置 RUN 模式选择器时下载程序以及更改参数。

3）S7-PLCSIM 支持 4 个累加器（类似于 S7-400 CPU）。在某些特殊情况下，同一程序在 S7-PLCSIM（带有 4 个累加器）中的运行方式可与 S7-300 CPU（仅带有两个累加器）中的运行方式不相同。

4）对视图对象所做的任何更改会立即更新存储单元中的内容。仿真 CPU 不会等到扫描开始或结束才更新所有已更改数据。

5）扫描模式选项允许选择 CPU 运行程序的方式：

-单次扫描；

-持续扫描。

6）可自动处理定时器，或手动输入值。还可以全部或单个复位定时器。

7）可以手动触发错误和中断 OB：

-OB40 ~ OB47（硬件中断）；

-OB70（I/O 冗余错误）；

-OB72（CPU 冗余错误）；

-OB73（通信冗余错误）；

-OB80（时间错误）；

-OB82（诊断中断）；

-OB83（插入/卸下中断）；

-OB85（程序顺序错误）；

-OB86（机架故障）。

8）过程映像和外设存储器：如果更改过程输入映像寄存器中的值，则 S7-PLCSIM 会立即将此值复制到该输入的 I/O 区。这样的话，如果在下一扫描开始时将 I/O 区输入值写入过程输入映像寄存器中，将不会丢失所需更改。相应地，如果更改了 I/O 区输出值，则会立即将此值复制到过程输出映像寄存器中。在仿真 CPU 中，当从 STEP 7 变量表中修改变量时，必须确保过程映像更新未重写或覆盖预期修改内容。

按照以下步骤为修改变量设置触发点：

1）对于输入，请选择“扫描周期开始”（Beginning of scan cycle）作为“修改的触发点”（Trigger Point for Modifying）。

2）对于输出，请选择“扫描周期结束”作为“修改的触发点”（Trigger Point for Modifying）。

2. 与“真正”PLC 的区别

仿真 PLC 不提供以下“真正”PLC 所具备的功能：

1）诊断缓冲区：S7-PLCSIM 不支持写入诊断缓冲区的所有错误消息。例如，在 CPU 或 EPROM 错误中无法仿真与故障电池相关的消息。但大多数的 I/O 和程序错误都可以进行仿真。

2）工作模式的切换不会将 I/O 更改为“安全”状态。

3）不支持函数模块（FM）。

4）不支持点对点通信（如同一机架上的两个 S7-400 CPU 之间的通信）。

5）S7-PLCSIM 不支持强制变量。

6）S7-PLCSIM 执行某些 SFB 和 SFC 的方式与真正 S7 PLC 的方式相同；至于其他方面，S7-PLCSIM 验证输入参数并返回有效输出，而带有物理 I/O 的真正 S7 PLC 所返回的输出不一定是有效输出；S7-PLCSIM 将多余部分视为 NOP。

7）无论要仿真哪种型号的 CPU，S7-PLCSIM 的本地数据的大小均定义为每个优先级 32KB。由于实际硬件中的本地数据组态可能与 S7-PLCSIM 中的不同，因此可能会出现下载到该硬件时被拒绝的情况。

8）S7-PLCSIM 不支持多值计算：S7-PLCSIM 无法使用多个 CPU 仿真 SIMATIC 站（多值计算）。

9）S7-PLCSIM 不支持 H 系统。

10）PLCSIM 不支持 PROFINET I/O。

I/O 的区别

S7-300 系列的大多数 CPU 均自动组态 I/O：将模块插入物理控制器后，CPU 会自动识别此模块。仿真 PLC 无法仿真自动组态功能。如果将程序从自动组态 I/O 的 S7-300 CPU 下载到 S7-PLCSIM 上，则系统数据中不包含 I/O 组态。因此，首先必须将带有已组态 I/O 模块的硬件配置下载到系统数据中，以便定义 CPU 应使哪个模块可用。

要执行此操作，请创建一个项目，然后组态 S7-300 CPU（其中未自动组态 I/O），例如 CPU 315-2DP、CPU 316-2DP 或 CPU 318-2。将此硬件配置下载到 S7-PLCSIM 中。然后可以下载任意 S7 项目中的程序块。应用这些 I/O 时不会出现错误。

3. 仿真查看窗口

S7-PLCSIM 的仿真查看窗口包含工作区、标题栏、状态栏以及 S7-PLCSIM 菜单和工具栏。S7-PLCSIM 布局即为您显示视图对象的位置。仿真查看窗口如图 5-23 所示。

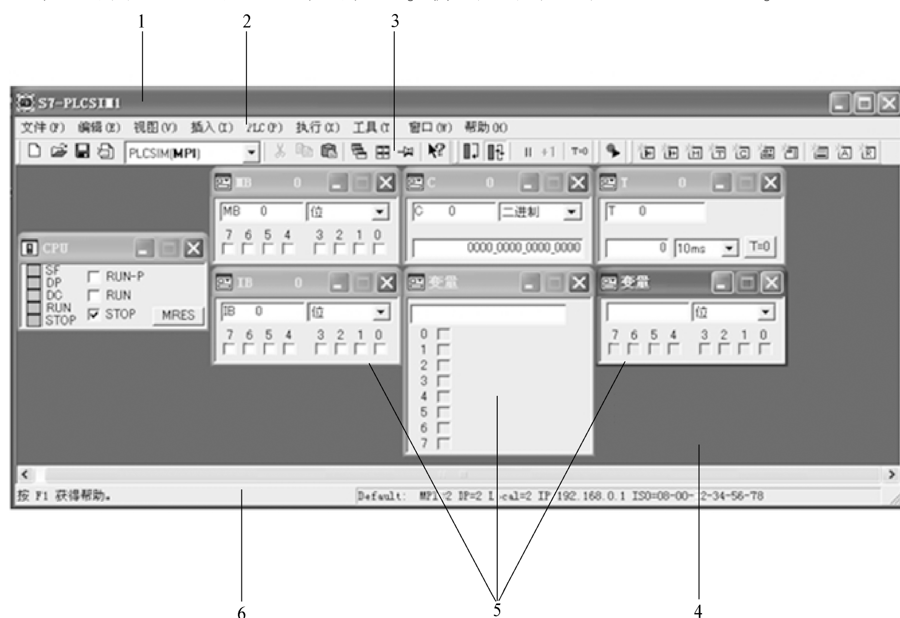


图 5-23　仿真查看窗口

1—标题栏　2—菜单栏　3—工具栏　4—工作区　5—视图对象　6—状态栏命令

4. 存储区

可以通过寻址存储器的特定区域来访问 S7 PLC 中的数据，这会执行特定功能。

存储区说明见表 5-1。

表 5-1 存储区说明

存 储 区	说 明	寻 址	S7-PLCSIM 范围
定时器	存储定时器	T	T0 ~ T2047
计数器	存储计数器	C	C0 ~ C2047
位存储器	存储在 STEP 7 用户程序中使用的数据	M	M 存储器为 131,072 位(16KB)
可寻址的 I/O	直接访问输入和输出模块 注:CPU 在各 CPU 扫描周期结束时更新外设输出	PI:外设输入 PQ:外设输出	I/O 存储器为 262,136 位(32KB)
过程映像(可组态;每隔一次扫描更新一次)	输入和输出的过程映像存储 注:CPU 在各 CPU 扫描周期开始时更新输入	I:输入 Q:输出	最大值:131,072 位(16KB) 默认设置:131,072 位(16KB)
本地数据(可组态)	供逻辑块(包括临时变量)使用的存储器	-/-	最大值:32KB 默认设置:32KB
数据块	数据块存储器	DB:数据块	最大数量 > 65534 最大长度:65570

5. 块

(1) 组织块(OB)

S7-PLCSIM 支持的 OB 见表 5-2。

表 5-2 S7-PLCSIM 支持的 OB

OB	说 明	OB	说 明
OB1	自由周期	OB81 *	电源错误
OB10 ~ OB17	日时钟中断	OB82	诊断中断
OB20 ~ OB23	延迟中断	OB83	插入/卸下模块中断
OB30 ~ OB38	周期性中断	OB84 *	CPU 硬件故障
OB40 ~ OB47	硬件中断	OB85	优先级错误
OB55 *	状态中断	OB86	机架故障
OB56 *	更新中断	OB87 *	通信错误
OB57 *	特定供应商的中断	OB88 *	处理中断
OB60 *	多处理器中断	OB90 *	背景 OB
OB61 * ~ OB64 *	同步循环中断	OB100	暖起动
OB65 *	工艺同步中断	OB101	热起动
OB70	I/O 冗余错误	OB102	冷起动
OB72	CPU 冗余错误	OB121	编程错误
OB73	通信错误	OB122	I/O 访问错误
OB80	超时错误		

注：无法调用标有星号（*）的 OB。

(2) 系统功能块（SFB）

S7-PLCSIM 支持的 SFB 见表 5-3。

表 5-3　S7-PLCSIM 支持的 SFB

SFB 编号	简　　称	SFB 编号	简　　称
SFB0	CTU	SFB20	STOP
SFB1	CTD	SFB22	STATUS
SFB2	CTUD	SFB23	USTATUS
SFB3	TP	SFB31	NOTIFY_8P
SFB4	TON	SFB32	DRUM
SFB5	TOF	SFB33	ALARM
SFB8	USEND	SFB34	ALARM_8
SFB9	URCV	SFB35	ALARM_8P
SFB12	BSEND	SFB36	NOTIFY
SFB13	BRCV	SFB37	AR_SEND
SFB14	GET	SFB52	RDREC
SFB15	PUT	SFB53	WRREC
SFB19	START	SFB54	RALRM

（3）系统功能（SFC）

S7-PLCSIM 支持的 SFC 见表 5-4。

表 5-4　S7-PLCSIM 支持的 SFC

SFC 编号	简　　称	SFC 编号	简　　称	SFC 编号	简　　称
SFC0	SET_CLK	SFC27	UPDAT_PO	SFC54	RD_DPARM
SFC1	READ_CLK	SFC28	SET_TINT	SFC55	WR_PARM
SFC2	SET_RTM	SFC29	CAN_TINT	SFC56	WR_DPARM
SFC3	CTRL_RTM	SFC30	ACT_TINT	SFC57	PARM_MOD
SFC4	READ_RTM	SFC31	QRY_TINT	SFC58	WR_REC
SFC5	GADR_LGC	SFC32	SRT_DINT	SFC59	RD_REC
SFC6	RD_SINFO	SFC33	CAN_DINT	SFC62	CONTROL
SFC9	EN_MSG	SFC34	QRY_DINT	SFC64	TIME_TCK
SFC10	DIS_MSG	SFC36	MSK_FLT	SFC78	OB_RT
SFC11	DPSYC_FR	SFC37	DMSK_FLT	SFC79	SET
SFC12	D_ACT_DP	SFC38	READ_ERR	SFC80	RSET
SFC13	DPNRM_DG	SFC39	DIS_IRT	SFC82	CREA_DBL
SFC14	DPRD_DAT	SFC40	EN_IRT	SFC83	READ_DBL
SFC15	DPWR_DAT	SFC41	DIS_AIRT	SFC84	WRIT_DBL
SFC17	ALARM_SQ	SFC42	EN_AIRT	SFC85	CREA_DB
SFC18	ALARM_S	SFC43	RE_TRIGR	SFC87	C_DIAG
SFC19	ALARM_SC	SFC44	REPL_VAL	SFC90	H_CTRL
SFC20	BLKMOV	SFC46	STP	SFC105	READ_SI
SFC21	FILL	SFC47	WAIT	SFC106	DEL_SI
SFC22	CREAT_DB	SFC49	LGC_GADR	SFC107	ALARM_DQ
SFC23	DEL_DB	SFC50	RD_LGADR	SFC108	ALARM_D
SFC24	TEST_DB	SFC51	RDSYSST		
SFC26	UPDAT_PI	SFC52	WR_USMSG		

二、起动仿真

要起动仿真，请进行如下操作：

1. 可以使用以下方法之一起动 S7-PLCSIM

1）打开 SIMATIC Manager，选择菜单命令“选项→仿真模块”（Options→Simulate Modules），打开 S7-PLCSIM。用户界面语言和助记键设置与 STEP 7 设置相同。

2）从 Windows“开始”菜单中，选择菜单命令“SIMATIC→STEP 7→S7-PLCSIM 仿真模块”

（SIMATIC→STEP 7→S7-PLCSIM Simulating Modules）。

打开 S7-PLCSIM。用户界面语言与 STEP 7 设置不相同。如果是第一次起动 S7/PLCSIM，则界面语言为英语。在后续起动中，S7-PLCSIM 将使用上次使用的语言打开。该设置是用户特定的设置。

仿真已起动。视图对象“PLC”已打开。此 PLC 必须处于初始状态。它具有以下属性和标准设置：

1）支持任何连接。
2）支持任何地址。
3）标准地址。
4）以上次使用的界面为基础组态界面。
5）可立即下载。

使用仿真 PLC 可自动建立任意新连接。下载的所有程序均会转入仿真 PLC。如果单击 SIMATIC Manager 上的“可访问节点”（Accessible Nodes）按钮，则将显示仿真 PLC 的节点地址。

S7-PLCSIM 会自动将“S7ONLINE 访问点”更改为仿真子网。在仿真期间，请勿通过“设置 PG/PC 接口”将此访问点更改为 S7-PLCSIM 未知的访问点。在结束仿真时，S7-PLCSIM 会将访问点更改回初始设置。

2. 设置 PG/PC 接口

在 S7-PLCSIM 的先前版本中，仅能通过 MPI 连接仿真 PLC。在 S7-PLCSIM 中，可以通过以下任一接口组态类型进行连接：

1）PLCSIM（ISO）。
2）PLCSIM（本地）。
3）PLCSIM（MPI）。
4）PLCSIM（PROFIBUS）。
5）PLCSIM（TCP/IP）。

接口组态类型见表 5-5。

表 5-5 接口组态类型

接口组态	连接类型	接口组态	连接类型
PLCSIM(ISO)	通过 MAC 地址	PLCSIM(PROFIBUS)	通过 PROFIBUS 接口
PLCSIM(本地)	通过虚拟底板总线/软总线	PLCSIM(TCP/IP)	通过 IP 地址
PLCSIM(MPI)	通过 MPI 接口		未知连接类别

说明：在 S7-PLCSIM 中，使用 MPI 接口的连接属于仿真 PLC 的默认设置。随后，仿真 PLC 将从上次使用的连接类别开始。

要设置 PG/PC 接口，请进行如下操作：

1）在 STEP 7 中组态硬件配置。
2）起动 S7-PLCSIM。
3）在“标准”（Standard）工具栏的下拉列表中，为虚拟 PLC 选择其中某个已组态的连接类别。

说明：在 SIMATIC Manager 中，在“标准”（Standard）工具栏的下拉列表中所作的更改将影响菜单命令“工具→设置 PG/PC 接口”（Tools→Setting the PG/PC interface）的功能。反过来 也是如此。

下拉条目中的颜色含义：

- 黑色-黑色

此颜色表示 CPU 支持此 PG/PC 接口。可通过此接口完全访问 CPU。

- 灰色-灰色

此颜色表示 CPU 不支持此 PG/PC 接口。无法通过此接口访问 CPU。可选择此接口。但不能访问 CPU。

- 黑色-灰色

使用多个有相同地址的 CPU 时会显示此颜色。它表示虽然 CPU 支持 PG/PG 接口，但目前无法通过此接口进行访问。

3. S7-PLCSIM 的多重背景

通过使用此新功能，可以同时仿真多个 CPU。

1）至少打开了一个 S7-PLCSIM 实例。

2）STEP 7 中组态网络地址与 S7-PLCSIM 中的网络地址一致，或 PLC 处于初始状态。

要同时仿真多个 CPU，请按照下列步骤操作：

1）打开一个新实例。

2）选择菜单命令“仿真→新 PLC”（Simulation→New PLC）。

新的仿真实例将以初始状态起动，将打开“CPU”视图对象。

使用多重背景时的具体下载方法如下：

所有默认 CPU 都具有相同的默认地址，下载过程中不会考虑此问题。如果一个默认 CPU 打开了多重背景，则会将 STEP 7 项目下载到标题栏中编号的实例中（例如：S7-PLCSIM2）。这就是实例编号。

如果打开了多重背景，关闭这些实例时，必须注意以下事项：

1）要关闭所有实例，请关闭 SIMATIC Manager。

2）要分别关闭各实例，请选择菜单命令“仿真→关闭”（Simulation→Close）。

（1）选择连接类型

要更改 S7-PLCSIM 中的 PG/PG 接口，请使用“标准”（Standard）工具栏中的下拉列表。根据仿真 CPU 的数量以及是否可以通过设定接口进行访问，下拉列表中的条目可能具有不同的颜色。只有同时仿真多个 CPU 时，才会显示下列颜色：

黑色-灰色

使用多个有相同地址的 CPU 时会显示此颜色。它表示虽然 CPU“1”支持 PG/PG 接口，但目前无法通过此接口进行访问。这是因为已选择 CPU“2”使用同一地址进行通信。要选择 CPU“1”进行通信，则需要再次选择该 PG/PC 接口。随后，CPU“1”的颜色会变为“黑色-黑色”。CPU“2”的颜色会变为“黑色-灰色”。

（2）支持的通信块

S7-PLCSIM V5.4 的多实例起动功能支持在 CPU 间进行通信。

通过使用此新功能，可以同时仿真多个 CPU。如果预先装载了相应的硬件配置，则不同的 CPU 可以相互进行通信。这要求涉及的 CPU 在同一子网中具有不同的地址。CPU 之间的通信支持下列通信块：

1）SFB8“USEND”。

2）SFB9“URCV”。

3）SFB12“BSEND”。

4）SFB13“BRCV”。

5）SFB15“PUT”。

6）SFB14“GET”。

7）SFB19“START”。

8）SFB20“STOP”。

9）SFB22“STATUS”。

10）SFB23“USTATUS”。

说明：S7-PLCSIM 不具有实时功能。通信过程中可能会存在时间响应限制。

4. 下载 STEP 7 项目

要求：

1）仿真已从 STEP 7 SIMATIC Manager 中起动。

2）已组态恰当的连接类型。

3）STEP 7 中的地址与 S7-PLCSIM 中的地址对应，或 PLC 处于初始状态。

要下载 STEP 7 项目，请按以下步骤操作：

1）导航至 SIMATIC Manager 中的站。

2）单击下载图标，或选择菜单命令“PLC→下载”（PLC→Download）。

块和硬件配置被下载至仿真 PLC 中。仿真系统采用已加载 CPU 的标识和所有已组态连接数据。状态栏提供已在硬件配置中组态的网络地址总览。仿真系统使用“MRES”功能复位至系统的初始状态。

说明：独立组态的 CP 无法仿真独立组态的 CP。

5. 仿真和监视

要仿真应用程序以及监视和控制应用程序，请进行如下操作：

1）打开 SIMATIC Manager。

2）打开 STEP 7 示例项目“ZEn01_09_STEP7_Zebra”。

3）单击起动仿真符号以应用 S7-PLCSIM。

4）下载示例项目。

5）在 S7-PLCSIM 中创建其他“视图对象”。可对仿真 PLC 中的数据进行监视。

-单击输入图标，或选择菜单命令“插入→输入变量”（Insert→Input Variable）。此视图对象显示 IB0（输入字节 0）。将数据格式设置为“位”。

-单击输出图标，或选择菜单命令“插入→输出变量”（Insert→Output Variable）以插入第二个视图对象 QB0（输出字节 0）。

-单击定时器图标，或选择菜单命令“插入→定时器”（Insert→Timer）三次以插入 3 个“定时器”视图对象。在对应文本框中输入 2、3 和 4（针对定时器 T2、T3 和 T4），每次输入后按“Enter”键（S7-PLCSIM 将为这 3 个定时器中的每个定时器填写符号名称）。

6）选择菜单命令“PLC→接通电源”（PLC→Power On）。

7）选择菜单命令“执行→扫描模式→持续扫描”（Execute→Scan Mode→Continuous Scan）。

8）选择菜单命令“执行→钥匙开关位置→RUN”（Execute→Key Switch Position→RUN）或 RUN-P。仿真 CPU 处于 RUN 模式。

9）单击 IB0 的位 0，以仿真打开输入 0.0。

10）监视对定时器的影响。

11）单击保存图标，或选择菜单命令“文件→PLC 另存为”（File→Save PLC As）以将当前状态的仿真 PLC 另存为一个新文件。

6. 在 STEP 7 中监视程序仿真

要求：

1）视图对象已创建。

2）示例项目“Zebra”已打开，同时已将站下载至 S7-PLCSIM 中。

要在 STEP 7 中对程序仿真进行监视，请按以下步骤操作：

1）单击在线图标，或菜单命令“视图→联机”（View→Online）。“联机”模式将被激活。

2）浏览至 ZEBRA 示例项目中的“块”（Blocks）对象。

3）打开函数 FC1。将调用“LAD/STL/FBD”应用程序。

4）将仿真 CPU 设置为 RUN 模式。

5）打开 IBO 的位 0。

6）选择“LAD/STL/FBD 编辑器”（LAD/STL/FBD Editor）中的菜单命令“调试→监视”（Debug→Monitor）。可监视对程序的影响。

7. 使用帮助

可以通过“帮助”（Help）菜单或采用以下方式访问 S7-PLCSIM 在线帮助：

1）要在 S7-PLCSIM 窗口中获取某一对象的帮助信息，请在工具栏上单击“帮助”（Help）按钮，然后单击此对象。

2）要获取任意对话框或错误消息的帮助，请单击此对话框或消息框中的“帮助”（Help）按钮，或者按“F1”。“帮助”（Help）窗口提供以下按钮、菜单命令以及选项卡：

帮助按钮

• 隐藏按钮/显示按钮：切换导航区的显示（“目录”（Table of Contents）、“索引”（Index）以及“搜索”（Search）选项卡）。要减小帮助窗口的整体大小，可以隐藏导航区；在准备查看新主题时，请单击“显示”（Show）按钮以恢复此导航区。

• 后退按钮：如果已查看了多个主题，则通过此按钮，可以后移至先前主题。

• 前进按钮：如果已查看了多个主题，则通过此按钮，可以前移至下一主题。

• 主页：打开定义为 S7-PLCSIM 在线帮助主页的网页。

• 打印按钮：允许向已安装的打印机发送所选主题或整本书。

帮助浏览器选项卡

• 目录选项卡：选择此选项卡以查看帮助系统中的目录。双击任意书本图标，以展开此书并查看书中包含的主题。

• 索引选项卡：选择此选项卡以查看按字母数字排序的帮助系统索引关键字列表。

• 搜索选项卡：选择此选项卡，然后输入希望查找的字词。双击列表中的主题以查看该主题。默认情况下，在主题中，将在字词出现的位置以高亮形式显示字词，以便于找到字词。在显示主题前，可以在关闭和打开高亮显示之间进行切换。使用“选项”（Options）按钮执行此操作。

三、S7-PLCSIM 的使用

STEP7 标准版并不包括 S7-PLCSIM 软件包及授权，需单独购买。STEP7 Professional 版包括了 S7-PLCSIIM 的软件包及授权，安装即可。在菜单 Options 中，可以激活 S7-PLCSIM，此时再进行上传/下载/监控等操作就是针对 S7-PLCSIM 了，而不会对真实 PLC 进行操作（不论 PLC 是否联机）。

1. S7-PLCSIM 调用

可以通过 STEP7 菜单“Options→Simulate Modules”，激活 S7-PLCSIM；或者通过点击工具栏

中的图标，来激活 S7-PLCSIM。

2. S7-PLCSIM 简单示例

（1）S7-PLCSIM 界面

图 5-24 为 S7-PLCSIM 工作界面。

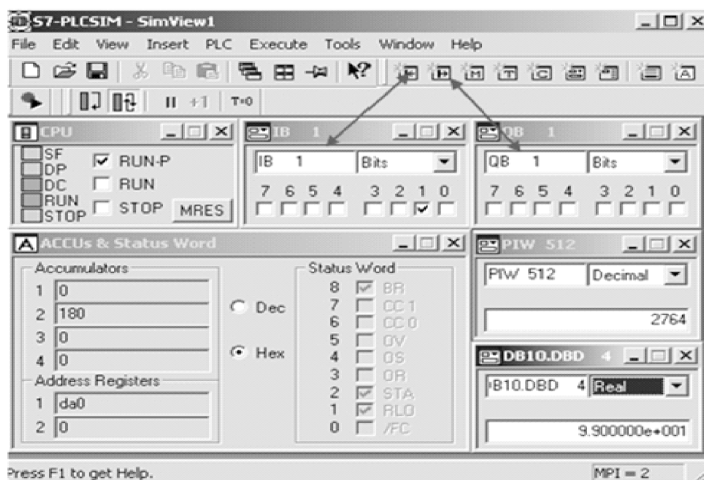


图 5-24 S7-PLCSIM 工作界面

（2）S7-PLCSIM 菜单

1）File 菜单：用户可以通过 S7-PLCSIM 菜单“File→Save PLC As”，将当前模拟的 PLC 存储为一个文件，下次使用时可以通过“File→Open PLC”直接打开此文件，而不需要下载，方便调试。对于 S7-PLCSIM V5.4 版本，可以在此设置多种下载方式，例如，MPI、DP、Ethernet。

2）View 菜单：用户可以通过“View→Accumulators/Block Registers/Stacks”来查看 PLC 内部的累加器/地址寄存器/状态字/堆栈资源。

3）Insert 菜单：用户可以通过“Insert→Input Variable”插入变量（输入/输出/中间寄存器/定时器/计数器/数据块）来模拟各种工况。

4）PLC 菜单：用户可以通过 PLC 菜单模拟真实 PLC 的上电/断电、内存复位操作以及修改 PLC 的 MPI 地址（S7-PLCSIM V5.4 版本以下）。

5）Execute 菜单（仅对部分内容作解释）：

① Key Switch Position：RUN 与 RUN-P 的区别，在 RUN 情况下，用户无法下载程序及修改 S7-PLCSIM 内部存储区；在 RUN-P 情况下，用户可以下载程序及修改 S7-PLCSIM 内部存储区，在两者中任意一种情况下，用户程序都可以正常运行。

② Startup Switch Position：用户可以选择当 S7-PLCSIM 由 STOP 模式转换到 RUN 模式时，执行的起动类型：Cold Start，操作系统将调用 OB102，用户程序从开始位置执行，存储在非保持区的用户数据被删除；Hot Start，操作系统将调用 OB101，并且用户程序从中断位置继续执行；Warm Start，操作系统将调用 OB100。

③ Scan Mode：Single Scan，S7-PLCSIM 特有的扫描模式，程序仅执行一个周期，当用户通过 Next Scan 操作时，S7-PLCSIM 执行下一个扫描周期；Continuous Scan，S7-PLCSIM 按照普通模式仿真真实 PLC 扫描模式。

④ Next Scan：当用户可以使能 S7-PLCSIM 执行下一个扫描周期。

⑤ Pause：在不影响输出的情况下，中断当前仿真的程序，注意在暂停的情况下，可能会导

致其他应用程序与 S7-PLCSIM 的超时或连接中断。

⑥ Automatic Timers：定时器自动运行。

⑦ Manual Timers：可以通过插入定时器窗口，手动设置定时器的值及时基。

⑧ Reset Timers：用户可以复位所有/部分的定时器。

⑨ Scan Cycle Monitoring：用户可以在此设置允许的最大程序执行时间，如果程序执行超过此时间，S7-PLCSIM 将进入停止状态。

6）Tools 菜单：

① Record/Playback：S7-PLCSIM 主要用于模拟工况，而即使一个简单的工况也可能是由一定时间段内的各种触发事件组成的。如果重复调试某个工况，而完全依赖于手工操作模拟是比较困难的。S7-PLCSIM 可以解决这个难题：编程人员可以将手工模拟过程录制成一个事件文件，针对不同的工况，可以录制不同的事件文件。选择不同的事件文件，即可模拟不同的工况，而不必一次又一次地去手动输入。

录制事件：此时操作者的每一步操作都会被记录下来。

录制事件界面如图 5-25 所示。

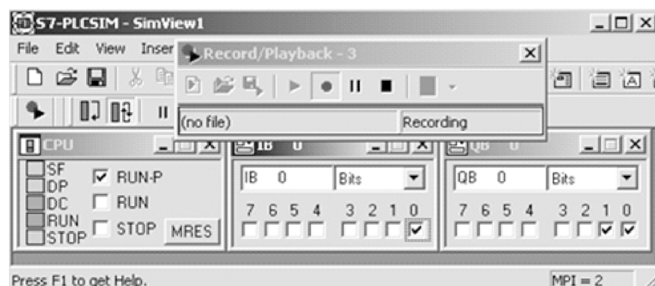


图 5-25　录制事件界面

事件回放：此时操作者的每一步操作会依次被重现（现在为第 2 个操作）。

事件回放界面如图 5-26 所示。

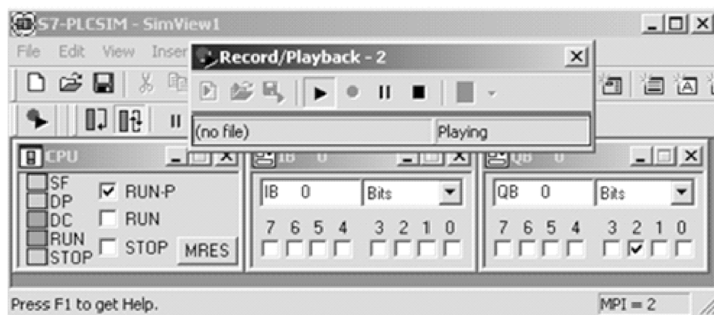


图 5-26　事件回放界面

② Options：在此菜单下 S7-PLCSIM 可以先使用 Attach Symbols，导入 STEP7 项目的符号表，然后在监控的情况下使用。

3. S7-PLCSIM 的常见问题

（1）问题：S7-PLCSIM 与在线连接的优先级

问题：当 S7-PLCSIM 已经运行，并且计算机已经与真实 PLC 有正确的编程连接方式，此时点击在线监控或者下载程序，STEP7 所访问的节点是 S7-PLCSIM 还是真实 PLC 呢？

解答：S7-PLCSIM 的优先级要高于真实 PLC 在线连接的优先级。也就是说，在 S7-PLCSIM 软件运行的情况下，所有的下载/上传/监控操作，都是针对 S7-PLCSIM 进行的，与真实 PLC 无关。有时计算机与真实 PLC 无法建立连接可能就是因为 S7-PLCSIM 正在运行，此时关闭 S7-PLCSIM 即可。

（2）问题：S7-PLCSIM 与 WinLC 的区别

问题：S7-PLCSIM 与 WinLC 有何区别？

解答：WinLC 的特性相当于真实 PLC 的特性，所以此问题请参考本文中 S7-PLCSIM 与真实 PLC 区别的章节。

（3）问题：无法调用 OB40

问题：为什么在 S7-PLCSIM 菜单中无法触发 OB40？

解答：S7-PLCSIM 仿真真实的 PLC，由于 OB40 与硬件组态关系密切，所以只有在下载了硬件组态后（或者 Block 文件夹下的 SDB 文件），在 S7-PLCSIM 菜单中才可以触发 OB40。

（4）问题：S7-PLCSIM 仿真通信程序

问题：S7-PLCSIM 是否可以仿真通信程序，例如：PTP 通信，以太网通信？

解答：S7-PLCSIM 无法仿真通信程序，此问题请参考本文中 S7-PLCSIM 与真实 PLC 区别的章节。

（5）问题：S7-PLCSIM 仿真定时器或定时中断功能

问题：S7-PLCSIM 是否可以仿真定时器或定时中断功能？

解答：S7-PLCSIM 的本质是一个在 Windows 环境下运行的应用程序，所以其执行状态与计算机的性能及系统资源使用状态都有着密切的联系。其仿真程序的扫描周期也实时受计算机负荷的影响，程序扫描周期可能会延长到几十个毫秒或者几百个毫秒。因此，当 S7 项目中的定时器时基定义非常小（例如 10ms）时，或者定时中断周期非常小（例如几个毫秒）时，S7-PLCSIM（受 Windows 运行机制及计算机性能影响）是无法在这么短的时间内完成相应功能的。对于真实的 PLC，由于其实时功能是由硬件来保证的，所以不存在上述问题（如果程序量比较大，程序扫描周期大于定时器的预设时间，这种情况下应当使用定时中断功能代替定时器的使用）。所以对于时序逻辑要求不严格的程序逻辑，可以使用 S7-PLCSIM 仿真；对于时序逻辑要求严格的程序逻辑，使用 S7-PLCSIM 仿真是不可靠的。

（6）问题：项目下载后，S7-PLCSIM 的 SF 点亮

问题：为什么项目下载后，S7-PLCSIM 的 SF 点亮，但程序仿真执行不受影响？

解答：这种情况多出现于向低版本的 S7-PLCSIM 软件下载了其无法识别的新硬件组态。用户升级 S7-PLCSIM 的软件版本即可。

四、故障排除提示

表 5-6 介绍了一些在使用 S7-PLCSIM 时可能遇到的问题。也列出了问题的可能原因及推荐的更正措施。

表 5-6　使用 S7-PLCSIM 时遇到的问题、原因及更正措施

问　　题	可能的原因及更正措施
程序无法下载到仿真 CPU 中	验证 CPU 是处于 STOP 模式还是 RUN-P 模式。如果仿真 CPU 处于 RUN 模式，则无法下载程序，除非已在 STEP 7 中组态 CiR（Configuration in RUN，运行中组态）元素。在 RUN 模式下，只有 CiR 对象可以下载到 S7-PLCSIM 中

（续）

问　　题	可能的原因及更正措施
程序无法下载到仿真 CPU 中	如果程序包含系统数据块（System Data Block，SDB），请验证 CPU 是否处于 STOP 模式。对于真正 CPU，仅当 CPU 处于 STOP 模式时才能下载 SDB 注：如果 CPU 视图对象处于 RUN-P 模式，则 STEP 7 会提示您切换到 STOP 模式，以便可下载硬件配置
	验证 CPU 和程序是否使用相同的节点地址和子网名称。为程序定义的节点地址必须与 CPU 的节点地址相匹配
S7-PLCSIM 应用程序不响应且显示“已锁定”	检查执行模式。仿真在暂停或处于单次扫描模式时可能会显示为已锁定
您输入一个外设变量但收到“地址无效”错误，尽管该地址值有效 -或- 您在程序中收到外设访问错误，尽管 S7-300 项目包含了正确的组态	只有 CPU 315-2DP、CPU 316-2DP 和 CPU 318-2 会下载 I/O 组态。如果从其他 S7-300 CPU 下载程序，则系统数据将不包含 I/O 组态。当尝试在 S7-PLCSIM 中访问外设 I/O 时，这会导致出错 要避免这些错误，首先应在系统数据中创建一个包含已组态 I/O 模块的硬件配置。这样，可以定义哪些 CPU 模块可用。要执行此操作，请创建一个项目，然后组态 S7-300CPU（其中未自动组态 I/O），例如 CPU 315-2DP、CPU 316-2DP 或 CPU 318-2。将此硬件配置下载到 S7-PLCSIM 中。然后可以下载任意 S7 程序中的程序块。应用这些 I/O 时不会出现错误
扫描因周期性中断而超期	仿真系统时，必须确保各个周期性中断 OB 的启动事件之间有足够的时间用于处理周期性中断。可能有必要相应地延长周期性中断的时间间隔

1. 参考信息

图标和菜单命令见表 5-7。

表 5-7　图标和菜单命令

图标	工具栏	菜 单 命 令	说　　明
		仿真	
	标准	文件 > 新建 PLC（File > New PLC）	生成一个具有处于原始状态的新 CPU 的新实例
	标准	文件 > 打开 PLC（File > Open PLC）	关闭当前仿真，并在同一实例中根据已保存的数据生成一个新 CPU
		文件 > 关闭 PLC（File > Close PLC）	关闭当前仿真，并在同一实例中生成一个处于原始状态的新 CPU
	标准	文件 > 保存 PLC（File > Save PLC）	保存当前仿真
		文件 > PLC 另存为（File > Save PLC As）	以新名称保存当前仿真
		文件 > 打开布局（File > Open Layout）	打开已保存的布局
		文件 > 关闭布局（File > Close Layout）	关闭当前布局
		文件 > 保存布局（File > Save Layout）	将当前排列保存为布局
		文件 > 布局另存为（File > Save Layout As）	以新名称保存当前布局
		文件 > 最近仿真（File > Recent Simulation）	打开最近的仿真
		文件 > 最近布局（File > Recent Layout）	打开最近的布局
		文件 > 退出（File > Exit）	关闭应用程序的所有窗口并退出应用程序
		编辑	
		编辑 > 撤消（Edit > Undo）	撤消上一次动作
	标准	编辑 > 剪切（Edit > Cut）	删除所选对象并将其保存到剪贴板上
	标准	编辑 > 复制（Edit > Cop/）	复制所选对象并将其保存到剪贴板上

（续）

图标	工具栏	菜单命令	说明
	标准	编辑 > 粘贴（Edit > Paste）	将剪贴板的内容插入到光标位置
		视图	
	插入对象	视图 > 累加器（View > Accumulators）	显示累加器 1 ~4 和状态字
	插入对象	视图 > 块寄存器（View > Block Registers）	显示地址寄存器和数据块寄存器
	插入对象	视图 > 堆栈（View > Stacks）	显示 MCR 堆栈和嵌套堆栈
		视图 > 工具栏（View > Toolbars）	显示特定的工具栏（开关）
		视图 > 状态栏（View > Status Bar）	显示状态栏（开/关）
	标准	视图 > 始终前置（View > Always On Top）	始终在最前面显示仿真
		插入	
	插入对象	插入 > 输入变量（Insert > Input Variable）	显示输入变量
	插入对象	插入 > 输出变量（Insert > Output Variable）	显示输出变量
	插入对象	插入 > 位存储器（Insert > Bit Memory）	显示位存储器
	插入对象	插入 > 定时器（Insert > Timer）	显示定时器
	插入对象	插入 > 计数器（Insert > Counter）	显示计数器
	插入对象	插入 > 通用（Insert > Generic）	显示数值画面
	插入对象	插入 > 垂直位（Insert > Vertical Bits）	显示字节
		目标系统	
		PLC > 上电（PLC > Power On）	打开 PLC
		PLC > 断电（PLC > Power Off）	关闭 PLC
		PLC > 清除/复位（PLC > Clear/Reset）	删除控制程序和变量存储器
		执行	
		执行 > 钥匙开关位置（Execute > Key Switch Position）	将 CPU 的钥匙开关置于所选模式
		执行 > 启动开关位置（Execute > Startup Switch Position）	设置启动开关位置
	CPU 模式	执行 > 扫描模式（Execute > Scan Mode）	设置模式
	CPU 模式	执行 > 下一扫描（Execute > Next Scan）	运行下一扫描
	CPU 模式	执行 > 暂停（Execute > Pause）	立即暂停程序
		执行 > 自动定时器（Execute > Automatic Timers）	将所有定时器都设置为自动模式
		执行 > 手动定时器（Execute > Manual Timers）	将所有定时器都设置为手动模式
	CPU 模式	执行 > 复位定时器（Execute > Reset Timers）	复位一个或所有定时器
		执行 > 触发错误 OB（Execute > Trigger Error OB）	触发错误 OB
		执行 > 扫描周期监视（Execute > Scan Cycle Monitoring）	用于设置和激活扫描周期监视时间
		工具	
	录制/回放文件	工具 > 录制/回放（Tools > Record/Playback）	录制或回放一系列事件
	标准	工具 > 选项 > 附加符号（Tools > Options > Attach Symbols）	搜索已下载程序的符号表

（续）

图标	工具栏	菜单命令	说明
		工具 > 选项 > 显示符号(Tools > Options > Show Symbols)	显示变量的符号
		工具 > 选项 > 引用数据(Tools > Options > Reference Data)	显示当前程序当前引用的数据
		工具 > 选项 > 符号表(Tools > Options > Symbol Table)	打开当前符号表
		窗口	
	标准	窗口 > 层叠(Window > Cascade)	排列所有已打开的窗口使它们层叠显示
	标准	窗口 > 按顺序平铺(Window > Tile Ordered)	按照逻辑顺序排列所有打开的窗口
		窗口 > 排列图标(Window > Arrange lcons)	沿窗口的底部边缘排列图标
		窗口 > 1、2、3...9(Window > 1,2,3...9)	激活已打开的视图对象
		帮助	
		帮助 > 目录(Help > Contents)	显示帮助主题的索引
		帮助 > 简介(Help > Introduction)	介绍此应用程序的功能范围
		帮助 > 入门指南(Help > Getting Started)	介绍使用此应用程序的必要步骤
		帮助 > 使用帮助(Help > Using Help)	显示有关使用帮助的信息
		帮助 > 关于(Help > About)	显示此应用程序当前版本的相关信息
	标准		在按钮、菜单和对话框上显示帮助

2. S7-PLCSIM 数值数据格式

S7-PLCSIM 支持的数值数据格式见表 5-8。

表 5-8　S7-PLCSIM 支持的数值数据格式

数值数据格式	大小	示例
位	位、字节	□ = 关闭，☑ = 打开
二进制	字节、字	1001_0011
十进制	字节、字、双字	232
Hex(十六进制)	字节、字、双字	9A
S7 格式	字节、字、双字	dw#16#9a2ff23
整型	字、双字	632，-2370
BCD(二进制编码的十进制)	字、双字	400
实型	双字	1.234567e+023
Char(字符)	字节、字、双字	"C"、"AB"
字符串	254 个字母数字字符	这是一个字符串
DT(DATE_AND_TIME)	8 个字节	2006-12-25-08:01:01 注：DT 数值数据格式不支持毫秒。 如果所有 8 个字节均为 0，则默认 DT 显示为：1999-11-30-00:00:00
S5TIME	WORD	3m5s00ms
日期	WORD	1998-06-18
定时器	双字	9h26m53s703ms
TOD	双字	9:26:53.702
滚动条：Dec	字节、字、双字	

（续）

数值数据格式	大　小	示　例
滚动条:Int	字、双字	MW 0 Slider:Int 0 Value
滚动条:实型	双字	MD 0 Slider:Rea 0.00000e+000 Value

第六节　调　试

一、用变量表调试

1. 调试的一般步骤

调试的一般步骤是先调试硬件，再调试软件程序。硬件调测可通过故障诊断工具诊断故障，观察 CPU 模块上的故障指示灯显示情况，也可通过变量表测试硬件。在变量表测试开始时，要先下载硬件组态数据和用户程序。

2. 变量表功能

由于变量表具有能够存储各种不同测试情况的优点，可易于进行保养和维护的测试和监控。变量表的可存储数目没有任何限制。其功能为：监视变量（在可编程设备/PC 上显示用户程序或 CPU 中单个变量的当前值）、修改变量（将固定值分配给用户程序或 CPU 的单个变量）、对外设输出赋值（固定值分配给处于 STOP 模式下的 CPU 的单个 I/O 输出）、强制变量（为用户程序或 CPU 的单个变量分配一个用户程序无法覆盖的固定值）、定义变量被监视或赋予新值的触发点和触发条件。

3. 监视修改变量的步骤

1）创建新变量表或打开一个已经存在的变量表。变量表窗口如图 5-27 所示。

2）编辑或检查变量表的内容。

3）使用菜单命令"PLC→连接到"，在当前变量表和所需的 CPU 之间建立在线连接。下载程序到 PLC。

4）使用菜单命令"变量→触发器"，选择合适的触发点并设置触发频率。

5）用菜单命令"变量→监视"和"变量→修改"，打开监视和修改功能。

6）使用菜单命令"表格→保存"或"表→另存为"来保存所完成的变量表，以便可以随时再次调用它。



图 5-27　变量表窗口

4. 变量表的生成

1）在管理器窗口生成变量表如图 5-28 所示。

2）在变量表编辑器中，可以用主菜单“表格→新建”生成一个新的变量表，如图 5-29 所示。



图 5-28　管理器窗口生成变量表

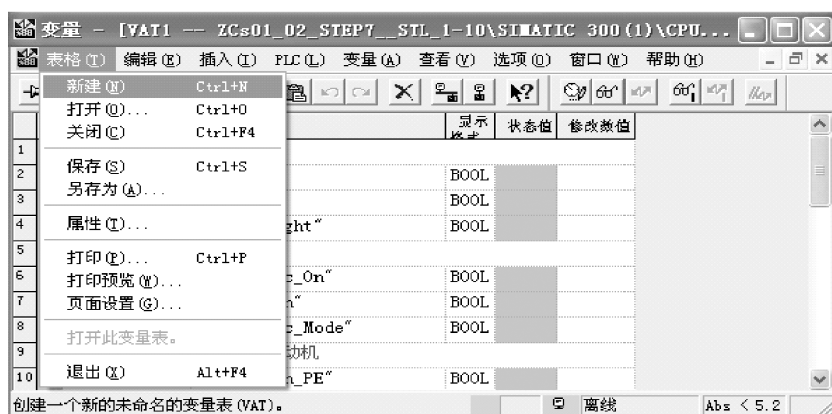


图 5-29　新建变量表

5. 变量表中的变量输入

变量的输入可以采取多种方式，如选择菜单命令“选项→符号表”，从符号表中复制符号，将它粘贴到变量表；输入希望通过地址进行修改的变量或作为符号的变量，可在“符号”栏或“地址”栏中输入符号和地址，该条目将会在对应的栏中自动写入，如图 5-30 所示。还可选择菜单命令插入，如图 5-31 所示，选择菜单命令“插入→行”增加变量；选择菜单命令“插入→变量范围”，弹出“变量的插入范围”对话框，在“起始地址”域中输入地址作为起始地址，在“编号”域中输入要插入的行数，从显示的列表选择要求的显示格式，点击“确定”按钮。

注意，在变量表中输入变量时，只能输入已经在符号表中定义过的那些符号而且必须完全按照符号表中的定义来输入符号，对于含特殊字符的符号名称必须包含在引号内（例如，“Motor. Off”）。

6. 变量表的使用

1）使用菜单命令“PLC→连接到→...”，建立到适当的 CPU 的连接，使之可监视或修改变量。

2）定义变量表触发方式。使用菜单命令“变量→触发器”来设置触发点和触发频率。选择相应的单选按钮，在对话框中定义用于监视变量的触发点和触发条件，如图 5-32 所示。

3）监视变量。使用菜单命令“变量→监视”启动或关闭监视处理，用于查看变量的状态值。



图 5-30 “符号”栏、“地址”栏中变量输入

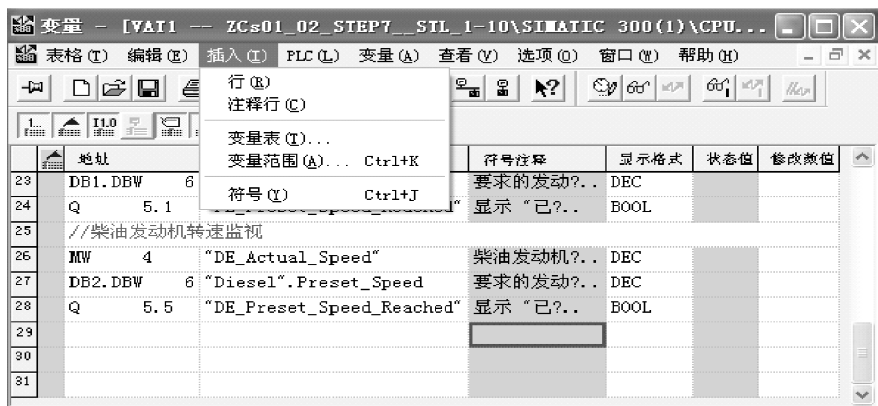


图 5-31 变量表编辑器

4）修改变量。使用菜单命令“变量→修改”，激活修改功能。根据设置的触发点和触发频率，用户程序为变量表中选择的变量应用修改值。如果设置的触发频率为“每次循环”，可以再次使用菜单命令“变量→修改”来关闭修改功能。需要注意的是，只有在变量表中修改开始时那些可见的地址，才能被修改。修改无法撤消（例如用“编辑→撤消”）。要在确保不会发生危险的状况下执行“修改”功能。当“修改”功能正在进行时，按下“ESC”键，将不做任何询问中止“修改”功能。

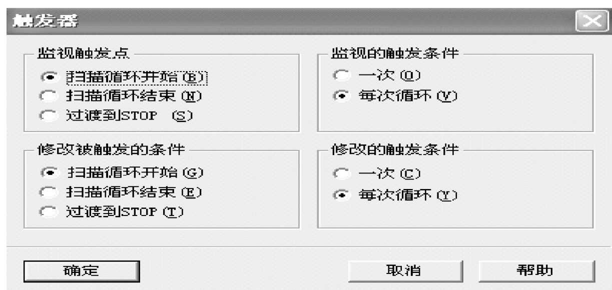


图 5-32 定义变量表触发方式

5）强制变量。只有当“强制值”窗口是激活的，才可以选择强制菜单命令。使用菜单命令“变量→显示强制值”来打开“强制值”窗口，在此窗口中显示所选择 CPU 的当前状态。如果 CPU 不支持强制，则不能选择“显示强制值”菜单命令。强制作业只能用菜单命令变量→停止强制来删除或终止。关闭强制值窗口或退出“监视和修改变量”应用程序不会删除强制作业。强制不能撤销（例如用“编辑→撤销”）。如果 CPU 不支持强制功能，与强制动作链接的变量菜单中的所有菜单命令都是取消激活的，如图 5-33 所示。一旦通过菜单命令“变量→强制操作”

给用户程序的单个变量分配了固定值，就不因程序执行而改变。如果使用菜单命令“变量→启用外设输出来”取消激活输出禁用，所有的强制输出模块都会输出它们的强制值。对于可支持强制变量功能的 CPU（例如，S7-400CPU），通过强制功能为变量分配固定的值，为用户程序设置特定的状况，然后以此来测试编写的功能。使用不正确的强制操作可能会造成人员伤亡和财产损失，对此操作要特别慎重。

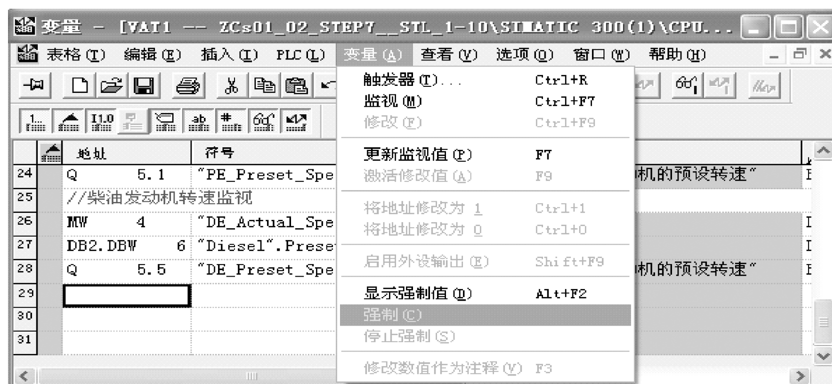


图 5-33　强制操作

二、用编程状态调试

1. 进入程序状态的条件和调试步骤

进入程序状态的条件：程序已编译下载到 CPU；打开逻辑块，用菜单命令“调试→监视”，进入在线监控状态；CPU 和用户程序正在运行。

调试的一般步骤是：定义程序状态的显示→选择调试的操作模式→定义调用环境（可选）→开/关调试。要设置断点，以单步模式来执行程序，必须设置测试操作模式（用菜单命令“调试→操作”设置）。这些测试功能不能用于过程操作模式。在用程序状态功能调试程序时，出现的错误可能会造成严重损害，对此要做好防范。因此一般不直接调用整个程序来调试，而是从调用体系最深的嵌套层的块开始，逐个地调用块，然后单独对其进行调试。例如，在 OB1 中调用它们，然后通过监视和修改变量，为块创建要测试的环境。

2. 程序状态显示

可以通过在程序编辑器中显示每条指令的程序状态（RLO、状态位）或相应寄存器的内容来调试程序。通过在程序编辑器窗口使用菜单命令“选项→自定义”对话框的“STL”标签里定义显示信息的范围，如图 5-34 所示。

用程序状态监视 STL 程序如图 5-35 所示，从光标选择的网络开始监视程序状态。右边窗口显示每条指令执行后的逻辑运算结果（RLO）和状态位 STA（Status）、累加器 1（STANDARD）、累加器 2（ACCU2）和状态字（STATUS…）。用菜单命令“选项（Options）→自定义（Customize）”打开的对话框分 STL 标签页选择需要监视的内容，用 LAD/FBD 标签页可以设置梯形图（LAD）和功能块图。

梯形图的状态显示如图 5-36 所示，在 LAD 和 FBD 中预设颜色，用绿色实线表示状态实现，蓝色点划线表示状态没有实现，黑色实线表示状态未知。线类型和颜色的预设值可以在菜单命令“选项→自定义”、“LAD/FBD”标签页下修改。程序状态的显示从选择的程序段开始，周期性更新。

梯形图中元素状态表示定义：触点的状态是如果地址具有值“1”代表实现，如果地址具有值“0”代表没有实现，如果地址值为未知则代表未知；具有启用输出（ENO）的元素的状态对应于将 ENO 输出值作为地址触点的状态，如果启用的输出没有连接，具有启用输出（ENO）的元素以黑色



图 5-34　定义状态域显示内容

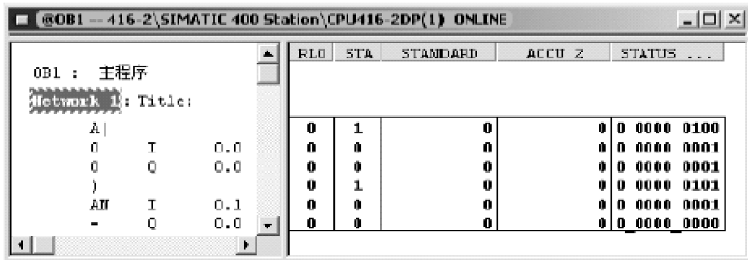


图 5-35　用程序状态监视 STL 程序

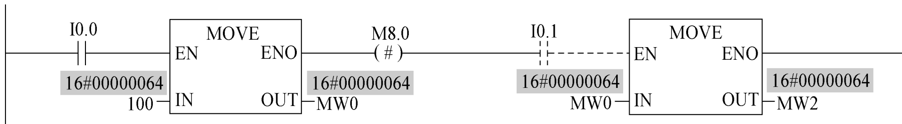


图 5-36　梯形图的状态显示

显示；具有 Q 输出的元素的状态对应于具有地址值触点的状态；如果调用后 BR 位被置位，那么 CALL 状态将实现；如果执行跳转（即如果跳转条件满足），那么跳转指令的状态将实现。

梯形图中线状态表示定义：如果未穿过线或如果线状态未知，则这些线是黑色的；从母线开始的线状态始终为“1”；从并联分支开始的线状态始终为“1”；如果元素前的线状态和元素状态均为“1”，那么元素后的线状态也将为“1”；如果在 NOT 前的线状态不为“1”，那么在 NOT 后的线状态为“1”（反之亦然）；如果满足在相交前至少有一条线的状态为“1”或在分支前的线状态为“1”，则线状态将在一些线相交后为“1”。

梯形图中参数状态表示定义：以粗体显示的参数值为当前值；以细体字显示的参数值来自前

一个周期或当前扫描周期不处理程序部分。

程序状态功能监视数据块。在数据视图（见图 5-37）中查看在线数据块。可以通过在线数据块或离线数据块激活显示。在这两种情况下，会显示 PLC 中的在线数据块的内容。在程序状态启动前，不得修改数据块。如果在线数据块和离线数据块的结构有所不同（声明），可以根据要求直接将离线数据块下载到 PLC 中。数据块必须位于“数据视图”中，以便在线值可以在“实际值”栏中显示。当状态激活时，不能切换到声明视图。更新时，在状态栏中可以看见绿色条，并显示工作模式，只能更新画面中可见的数据块部分。数值以相应数据类型的格式显示；格式不能改变。在程序状态结束后，“实际值”栏再次显示在程序状态之前有效的内容。不能将更新后的在线值传送到离线数据块中。所有基本数据类型都在共享数据块中更新，也可在所有的实例数据块的声明（输入/输出/输入-输出/静态）中更新。有些数据类型不能更新，如参数类型、DATE_AND_TIME 和字符串等。

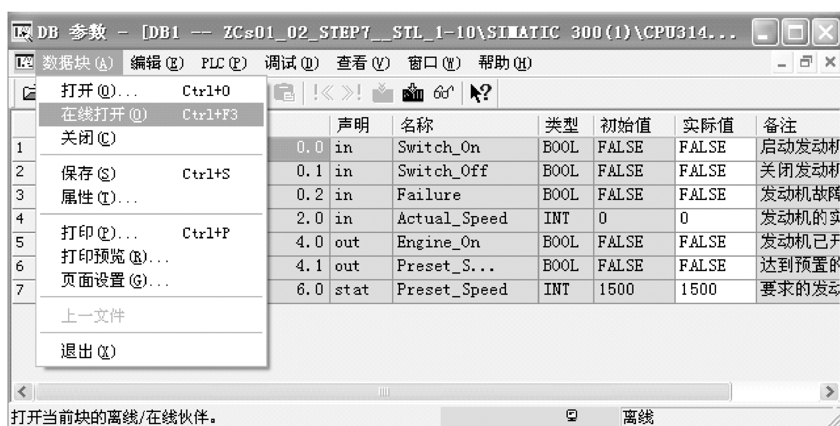


图 5-37　数据视图

3. 单步与断点调试

只能在程序编辑器中的语句表编程模式下进行单步/断点的设置和执行工作。在单步/断点模式下调试时，进行逐个语句地执行程序（单步）和设置断点的操作。进入单步/断点模式需满足以下几个条件。

1）只能在语句表中使用单步和断点功能，对于梯形图或功能块图中的块，必须使用菜单命令“查看→STL”来改变视图。

2）执行菜单命令“选项→自定义”，在对话框中选择 STL 标签页，激活“立即激活新断点”选项。

3）必须设置测试操作模式，单步模式测试不能在操作模式下进行，用菜单命令“调试→操作”使 CPU 工作在测试（Test）模式。

4）不得保护块，不得在编辑器中改变打开的块，必须在线打开被调试的块。

5）设置断点时不能起动程序状态（Monitor）功能。

6）STL 程序中有断点的行、调用块的参数所在的行、空的行或注释行不能设置断点。

在“调试”菜单中可以找到能用来设置、激活或删除断点的菜单命令，如图 5-38 所示。也可以使用断点工具栏中的图标选择那些菜单命令。使用菜单命令“查看→断点条”来显示断点工具栏。在满足单步/断点设置条件时，在语句表中设置断点，将光标置于要设置断点的指令行，在 STOP 或 RUN-P 模式下执行菜单命令“调试→设置断点”，语句行用空心圆圈标记断点，该指令行左侧会出现一小圆，如图 5-39 所示，表明断点设置成功，同时弹出可移动的显示寄存器的小窗口，

使用菜单命令“查看→PLC 寄存器”可以打开和关闭该窗口。单步模式一次只执行一条指令。



图 5-38 单步/断点操作设置

其他的单步/断点菜单指令有“调试→删除断点”（逐个删除断点），或者可以使用菜单命令“调试→删除所有断点”（删除所有断点）、“调试→断点激活”（激活后断点用实心圆圈标记）、“调试→显示下一个断点”（光标跳转到所选择的下一个断点，而无需处理块）、“调试→恢复运行”（继续运行程序直到下一个断点）、“调试→执行下一个命令”（在单步模式下进行测试）、“调试→执行调用”（跳转到块。可以继续在那里以单步模式测试，也可以设置断点。在块结束处，程序跳转回块调用后的下一个语句）。

OB1 : 主程序

Network 1 : 起保停电路

```
A(
O    I    0.0
O    Q    4.0
|
AN   I    0.1
=    Q    4.0
```

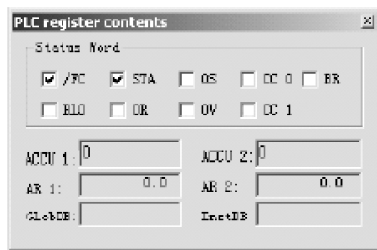


图 5-39 断点及断点处寄存器的内容

第七节 故 障 诊 断

一、故障诊断的基本方法

1. 定义诊断符号

诊断符号用来形象直观地检测故障。如果没有出现故障，那么所显示的模块类型符号上不带附加的诊断符号。如果模块有诊断信息，显示模块带附加的诊断符号，或以较低的对比度显示模块符号。诊断符号如图 5-40 所示。

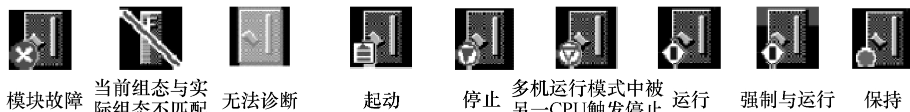


图 5-40 诊断符号

2. 硬件诊断和故障检测

通过出现的诊断符号，可以检查是否有可供模块使用的诊断消息。诊断符号说明了相应模块的状态，而且也说明了 CPU 工作模式。当调用功能（见图 5-41）“诊断硬件”后，诊断符号将会显示在在线视图以及快速视图（默认设置）或诊断视图的项目窗口中。双击快速视图或诊断视图中的诊断符号，可起动“模块信息”应用程序来显示详细的诊断信息。

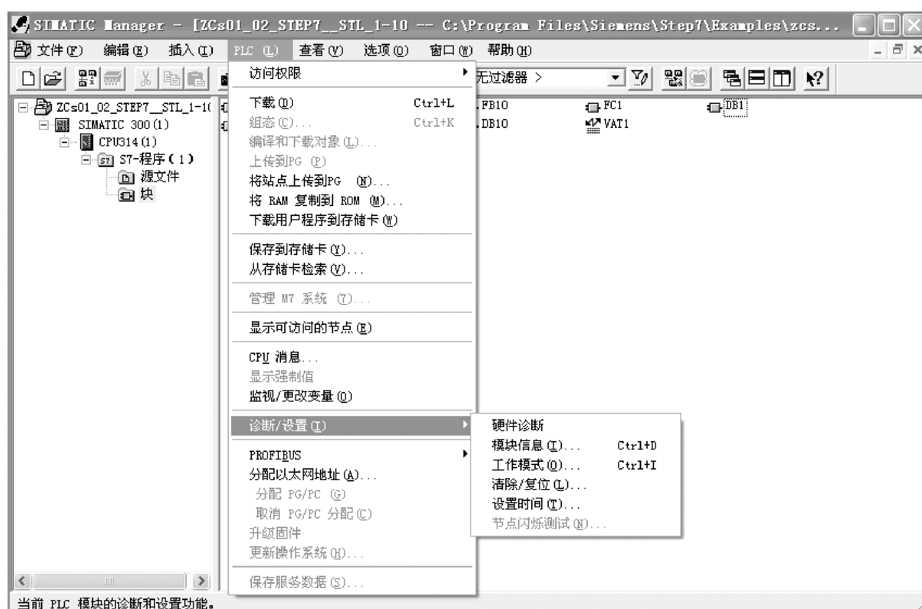


图 5-41　诊断设置

3. 故障定位方法

1）使用菜单命令“查看→在线”打开项目的在线窗口。

2）打开所有的站，以便在其中组态的可编程模块均为可见。看是否有 CPU 正在显示诊断符号，其指示了错误或故障。按“F1”键打开解释诊断符号的帮助页面。

3）选择要检查的站。

4）选择菜单命令“PLC→诊断/设置→模块信息”以显示该站中 CPU 的模块信息。

5）选择菜单命令“PLC→诊断/设置→诊断硬件”以显示该站中 CPU 和故障模块的“快速视图”。快速视图的显示已设置为默认值（菜单命令“选项→自定义”，“视图”标签）。

6）选择快速视图中的故障模块。

7）点击“模块信息”按钮以获取关于该模块的信息。

8）点击快速视图中的“在线打开站”按钮，以显示诊断视图。诊断视图包括了按照其插槽顺序排列在站中的所有模块。

9）双击诊断视图中的模块，以便显示模块信息。采用该方式，也可获得那些没有故障因而没有显示在快速视图中的模块的信息。

一旦获得所需要的诊断信息，即可停止上述定位故障行动，不必执行所有的这些步骤。

二、用快速视图和诊断视图诊断故障

1. 调用快速视图诊断故障

快速视图提供一种使用“诊断硬件”的快捷方式，其中的信息量少于在 HW Config 的诊断视图中的详细显示信息。当调用“诊断硬件”功能时，快速视图作为默认显示（见图 5-42）被打开。

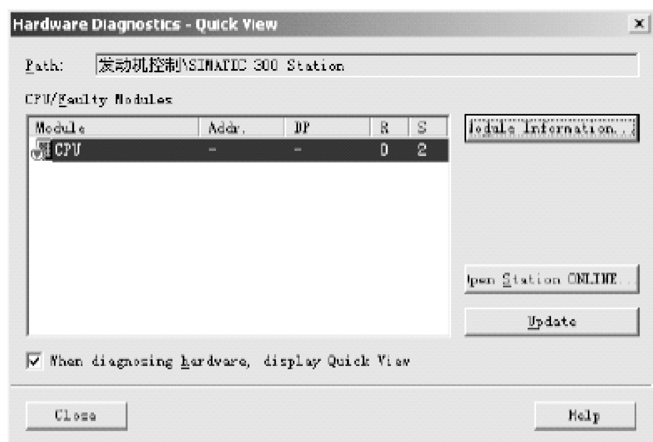


图 5-42　快速视图

在 SIMATIC 管理器中，使用菜单命令“选项→自定义→查看”对话框要激活“在硬件诊断期间显示快速视图”，如图 5-43 所示。

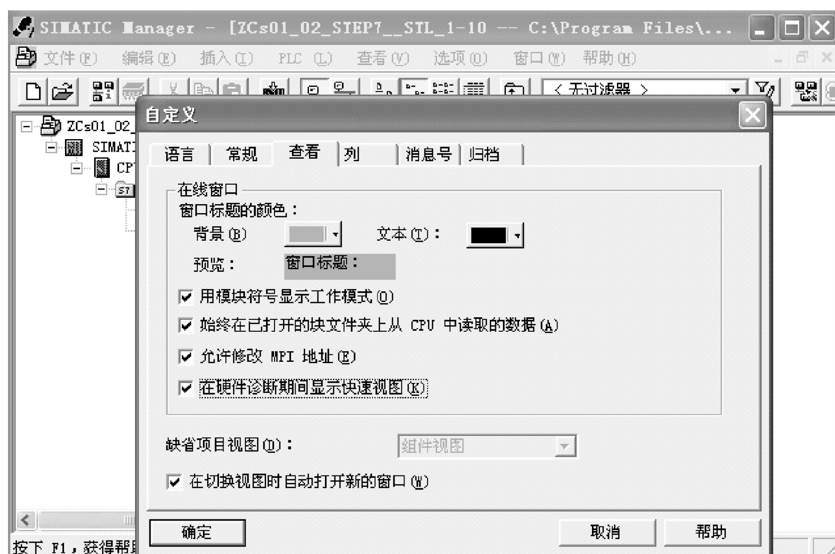


图 5-43　激活“在硬件诊断期间显示快速视图”

在 SIMATIC 管理器中，使用菜单命令“PLC→诊断/设置→诊断硬件”来调用该功能。可按如下方式使用该菜单命令：如果选择了一个模块或 S7/M7 程序，则在项目的在线窗口中；如果在“可访问节点”窗口中选择一个节点（“MPI = . . . ”），则该条目属于 CPU。从所显示的组态表中，可以选择希望显示其模块信息的模块。

快速视图中会显示如下信息：在线连接到 CPU 的数据和诊断符号、被 CPU 检测出故障的模块的诊断符号（例如，诊断中断、I/O 访问错误）和模块类型和模块地址（机架、插槽、具有站编号的 DP 主站系统）。

由于 STEP 7 必须查找并显示数据，故显示诊断视图所需的时间较长。因此，起动“诊断硬件”应用（默认设置）时，显示快速视图往往更有利。如有必要，可通过“在线打开站”按钮，打开诊断视图。

2. 调用诊断视图

在快速视图中使用“在线打开站”按钮显示诊断视图，可以打开一个诊断视图对话框，该

对话框与快速视图不同，包含整个站的图形总览以及组态信息。它侧重于在“CPU/故障模块”列表中高亮显示的模块。

在 SIMATIC 管理器的项目视图中，使用菜单命令“视图→在线”，建立到 PLC 的在线连接。双击打开选择的站。然后打开其中的“硬件”对象也可打开诊断视图。

与快速视图相比，诊断视图显示在线可用的整个站组态。整个站组态包含：机架配置、所有已组态模块的诊断符号（从这些诊断符号中，可以读取每个模块的状态，如果是 CPU 模块，则可读取工作模式）、组态的模块类型、订货号、地址详细资料以及注释。诊断视图中的附加诊断选项可通过双击模块，显示该模块的工作模式。

三、调用模块信息诊断故障

1. 模块信息窗口的打开

模块信息窗口的打开有以下途径。

（1）在 SIMATIC 管理器的项目窗口中打开

其方法是打开项目，选择一个站，然后双击打开该站，在站中选择一个模块或“S7 程序”文件夹，选择菜单命令“PLC→诊断/设置→模块信息”。

（2）在 SIMATIC 管理器的“可访问节点”窗口中打开

在 SIMATIC 管理器中使用菜单命令“PLC→显示”可访问节点，打开“可访问节点”窗口，在“可访问节点”窗口中选择一个节点（注意在“可访问节点”窗口中，只有具有本身节点地址的模块像以太网、MPI 或 PROFIBUS 地址才可见），然后选择菜单命令“PLC→诊断/设置→模块信息”。

在两种情况下，都显示“模块信息”对话框。根据模块的诊断能力，在“模块信息”对话框中显示不同数目的标签。

2. 在停止模式下诊断

在用户程序处理期间发生 CPU 不知因何原因进入 STOP 模式，确定 CPU 为何进入“停止”模式，需采取的操作是在 STOP 状态建立与 CPU 的在线连接，选择已进入停止模式的 CPU，执行菜单命令“PLC→诊断/设置→模块信息”打开模块信息对话框，选择“诊断缓冲区”标签。可以从诊断缓冲区的最后一个条目（亦即最上面一项）确定停止原因，如图 5-44 所示。

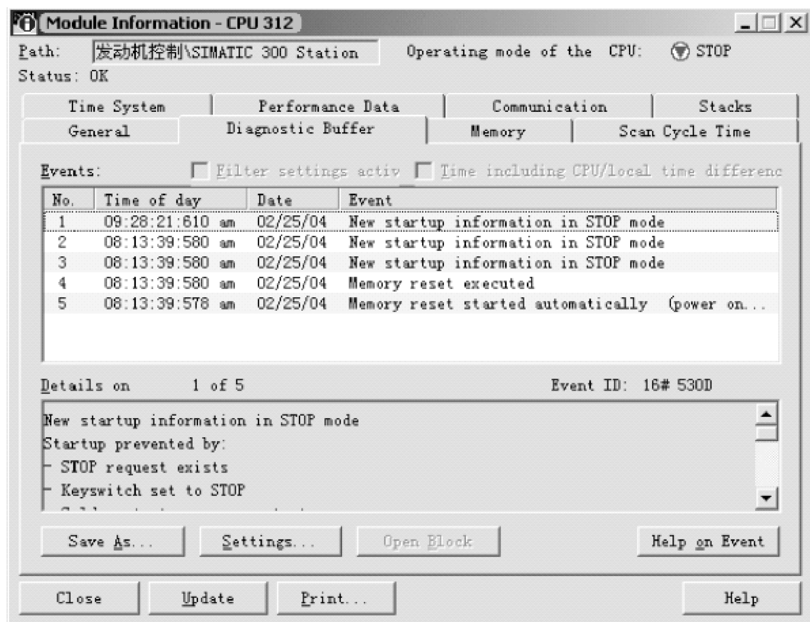


图 5-44　CPU 模块的在线模块信息窗口

第八节　显示参考数据

一、参考数据的生成与显示

STEP 7 的参考数据功能帮助用户创建参考数据并为其赋值，使用户程序的调试和修改更便捷、容易。在作为整个用户程序的一个概述、作为进行修改和测试的基础、为了补充程序文档这三种情况下可使用参考数据。

1. 生成参考数据

在 SIMATIC 管理器中，选择希望为其生成参考数据的块文件夹，在 SIMATIC 管理器中，执行菜单命令“选项→参考数据→生成”。在生成参考数据前，计算机检查是否有任何可用的参考数据，如果有，则检查数据是否是当前的。如果参考数据可用，则说明它们已经产生。如果可用的参考数据不是当前数据，则可以选择是否刷新参考数据或者是否再次完全生成它们，如图 5-45 所示。

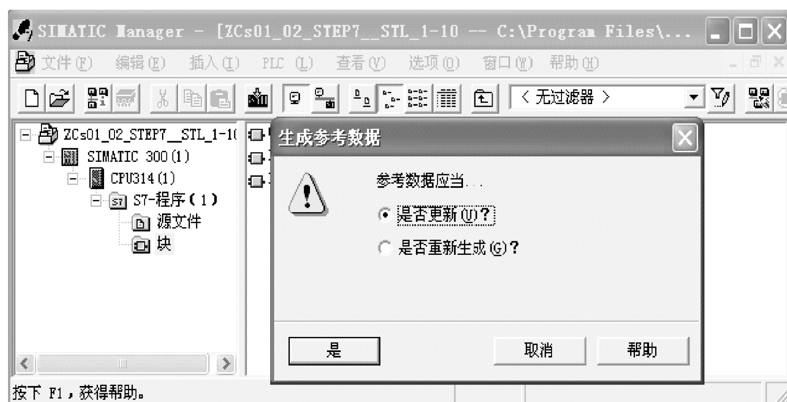


图 5-45　生成参考数据窗口

2. 显示参考数据

使用菜单命令“选项→参考数据→显示”可以显示参考数据。在显示参考数据前，进行检查以确定是否存在参考数据，以及存在的参考数据是否是当前的。如果不存在参考数据，则生成它们。如果存在不完整的参考数据，将显示一个对话框，提醒参考数据不一致。然后可以决定是否要刷新参考数据以及刷新到什么程度。为了刷新参考数据，需要对块进行重新编译。调用合适的编译器以编译每个块。使用菜单命令“视图→更新”可以刷新已显示在激活窗口中的参考数据的视图。

在“视图”菜单中可选择显示“交叉参考”、“赋值”、“程序结构”、“未使用的符号”、“不带符号的地址”，相互之间可以切换选择显示，如图 5-46 所示。

使用菜单命令“窗口→新建窗口”会生成新的参考数据窗口，同时打开一个 S7 用户程序的多个列表，如图 5-47 所示。

二、交叉参考表

交叉参考表（见图 5-48）给出了 S7 用户程序中关于地址使用的概况。窗口的每一行对应于交叉引用表的一个条目。当显示参考数据时，交叉参考表显示的是默认视图。可以改变此默认设置。使用过滤功能可以方便地查找特定的地址和符号。显示交叉参考表时，将获得存储区域输入（I）、输出（Q）、位存储区（M）、定时器（T）、计数器（C）、功能块（FB）、功能（FC）、系统功能块（SFB）、系统功能（SFC）、I/O（P）和数据块（DB）的地址列表，显示了它们在 S7



图 5-46　参考数据显示选择窗口



图 5-47　参考数据显示多个列表

用户程序中的地址（绝对地址或符号）和用途等的使用情况。交叉参考表默认为按照存储器区域排序。如果用鼠标点击栏目标题，可以按照默认排序标准对条目进行排序。

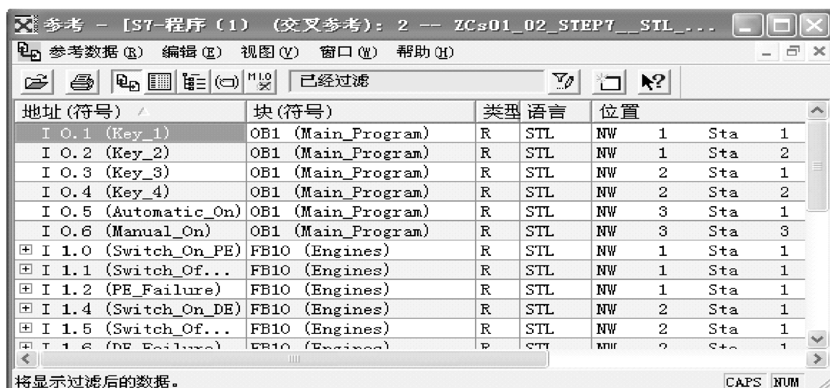


图 5-48　交叉参考表

三、程序结构

程序结构既显示 S7 用户程序中块的调用层级，同时又概要给出了所用的块、它们的从属关系和它们的局部数据要求，如图 5-49 所示。



图 5-49 程序结构显示

在“生成参考数据”窗口中使用菜单命令视图→过滤器，可以打开带选项卡的对话框。在“程序结构”标签页中，可以设置如何显示程序结构。可以在调用结构和从属性结构两者之间选择，选择界面如图 5-50 所示。

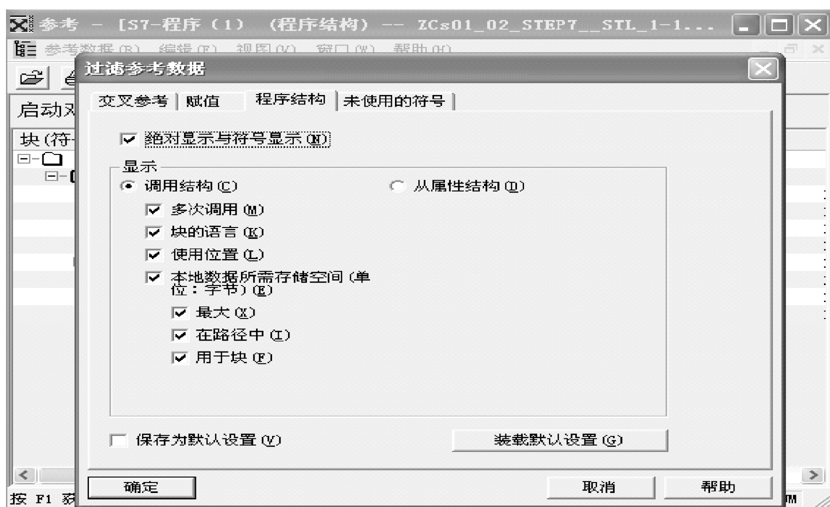


图 5-50 程序结构显示设置

四、赋值表

赋值表（见图 5-51）显示哪些地址已分配到用户程序中。用于用户程序中进行故障诊断或修改。I/Q/M 赋值表描述了存储器区域输入（I）、输出（Q）、位存取区（M）、定时器（T）和计数器（Z）的哪个字节的哪个位的使用情况。I/Q/M 赋值表显示在工作窗口中。每一行包含存储器区的一个字节，该字节的 8 个位根据它们的访问进行编码。它也指出是字节、字还是双字的访问。

I/Q/M 表中的标识中白色背景表示没有访问该地址，未赋值；X 表示直接访问地址，蓝色背景表示间接访问地址（字节、字或双字访问）。

五、未使用的符号

未使用的符号（见图 5-52）包括在符号表中定义的符号和存放参考数据的用户程序段中未使用的符号。在“生成参考数据”窗口中使用菜单命令“视图→未使用的符号”，可以通过点击栏目标题对条目排序。还可以从列表中删除不再需要的符号。为此，在列表中选择符号，然后执行“删除符号”



图 5-51　赋值表

功能。窗口的每一行对应于列表的一个条目。每行包括地址、符号、数据类型和注释。



图 5-52　未使用的符号

六、不带符号的地址

不带符号的地址如图 5-53 所示。在“生成参考数据”窗口中执行菜单命令视图→不带符号的地址，显示没有在符号表中定义但已在用户程序中使用的绝对地址的地址列表。工作窗口的标题栏显示列表所属的用户程序的名称。行包括地址以及地址在用户程序中使用的定时器的编号。条目按照地址存储。在列表中选择地址，然后执行“编辑符号”功能，可将名称分配给没有符号的地址。



图 5-53　不带符号的地址

第六章　S7-300/400用户程序结构与编程

本章主要了解用户程序的基本结构，掌握功能与功能块的使用方法，了解数据块与数据结构的相关知识，掌握多重背景的编程方法，了解组织块与中断处理的概念。

第一节　用户程序的基本结构

PLC 的 CPU 中运行的程序包括操作系统和用户程序，操作系统用来组织与特定控制任务无关的功能，例如处理 PLC 的重起、更新输入/输出过程映像表、调用用户程序、采集和处理中断、识别错误并进行错误处理、管理存储区和处理通信等。用户程序则由用户在 STEP7 中创建，并下载到 CPU 中。它包含处理特定的自动化任务所需要的所有功能，例如确定 CPU 重起或热重起的条件，处理过程数据，响应中断和处理程序正常运行中的干扰等。

用户程序可分为 3 个程序分区：主程序、子程序（可选）和中断程序（可选）。

主程序（OB1）：用户程序的主体。CPU 在每个扫描周期都要执行一次主程序指令。

子程序：程序的可选部分，只有当主程序调用时，才能够执行。合理使用子程序，可以优化程序结构，减少扫描时间。

中断程序：程序的可选部分，只有当中断事件发生时，才能够执行。中断程序可在扫描周期的任意点执行。

一、用户程序中的块

STEP7 编程软件允许用户将编写的程序和程序所需的数据放置在块中，使用户程序结构化，易于程序修改、查错和调试。块结构显著地增加了 PLC 程序的组织透明性、可理解性和易维护性。各种块的简要介绍见表 6-1。

表 6-1　用户程序中的块

块	功 能 简 介
组织块(OB)	决定用户程序的结构
系统功能块(SFB)和系统功能(SF)	集成在 CPU 模块中,通过调用 SFB 或 SFC,可以访问一些重要的系统功能
功能块(FB)	用户可以自行编程的带有存储区的块
功能(FC)	包含用户经常使用的功能的子程序
背景数据块(DI)	调用 FB 和 SFB 时,背景数据块与块关联,并在编译过程中自动创建
共享数据块(DB)	用于存储用户数据的数据区域,供所有的块共享

1. 组织块（OB）

组织块是操作系统与用户程序之间的接口，它们由操作系统调用，用于控制循环和中断程序的执行以及 PLC 的起动和错误处理等。组织块根据操作系统调用的条件（如时间中断、报警中断等），可以分成不同的类型，每种类型有不同的优先级，高优先级的 OB 可以中断低优先级的 OB。当 OB 起动时，提供触发它的初始化起动事件的详细信息，这些信息可以在用户程序中使用。

OB1 是主程序循环块，用于循环处理，操作系统在每一次循环中调用一次组织块 OB1。一个循环周期分为输入、程序的执行、输出和其他任务，例如下载、删除块、接收和发送全局数据等。根据过程控制的复杂程度，可将所有程序放入 OB1 中进行线性编程，或将程序用不同的逻

辑块加以结构化，通过 OB1 调用这些逻辑块，并允许块间的相互调用。这样可以把一个复杂的自动化任务分解为能够反映过程的工艺、功能或可以反复使用的小任务，使控制变得更加容易。

块的调用指令中止当前块的运行调用，然后执行被调用块的所有指令，当前正在执行的块在当前语句执行完后被停止执行（被中断），操作系统将会调用一个分配给该事件的组织块。该组织块执行完后，被中断的块将从断点处继续执行。

2. 局域数据

生成逻辑块（OB、FC、FB）时可以声明临时局域数据。这些数据是临时的，退出逻辑块时不保留临时局域数据。它们又是一些局域（Local，或称局部）数据，只能在生成它们的逻辑块内使用。CPU 按优先级划分局域数据区，同一优先级的块共用一片局域数据区。可以用 STEP7 改变 S7-400 每个优先级的局域数据的数量。

3. 功能（FC）与功能块（FB）

功能（FC）是用户编写的没有固定的存储区的块，其临时变量存储在局域数据堆栈中，功能执行结束后，这些数据就丢失了。利用共享数据区可以存储那些在 FC 执行结束后需要保存的数据，由于 FC 没有自己的数据存储区，所以不能为 FC 的局域数据分配初始值。

调用 FC 和 FB 时用实参（实际参数）代替形参（形式参数）。形参是实参在逻辑块中的名称，FC 不需要背景数据块。FC 和 FB 用输入（IN）、输出（OUT）和输入/输出（IN/OUT）参数做指针，指向调用它的逻辑块提供的实参。另外，FC 可以为调用它的块提供数据类型为 RETURN 的返回值。

功能块（FB）是用户编写的具有自己存储区域（背景数据块）的块，每次调用 FB 时需要提供各种类型的数据给 FB，FB 也要返回变量给调用它的块。这些数据以静态变量（STAT）的形式存放在指定的背景数据块（DI）中，临时变量 TEMP 存储在局域数据堆栈中。

调用 FB 或 SFB 时，必须指定 DI 的编号，调用时 DI 被自动打开。在编译 FB 或 SFB 时，系统会自动生成背景数据块中的数据。用户可以在用户程序中或通过 HMI（人机接口）来访问这些背景数据。

可以在 FB 的变量声明表中给形参赋初值，它们被自动写入相应的背景数据块中。在调用块时，CPU 将实参分配给形参的值存储在 DI 中。如果调用块时没有提供实参，将使用上一次存储在 DI 中的参数。

4. 数据块（DB）

数据块（DB）是用于存放执行用户程序时所需的变量数据的数据区。与逻辑块不同，在 DB 中没有 STEP7 的指令，STEP7 按数据生成的顺序自动地为 DB 中的变量分配地址。DB 分为共享数据块和背景数据块，其最大容量与 CPU 型号有关。

（1）共享数据块

共享数据块存储的是全局数据，所有的 FB、FC 或 OB（统称为逻辑块）都可以从共享数据块中读取数据，或将数据写入共享数据块。CPU 可以同时打开一个共享数据块和一个背景数据块。如果某个逻辑块被调用，它可以使用它的临时局域数据区（即 L 堆栈）。逻辑块执行结束后，其局域数据区中的数据丢失，但是共享数据块中的数据不会被删除。

（2）背景数据块

背景数据块中的数据是自动生成的，它们是功能块的变量声明表中的数据（不包括临时变量 TEMP）。背景数据块用于传递参数，FB 的实参和静态数据存储在背景数据块中。调用功能块时，应同时指定背景数据块的编号或符号，背景数据块只能被指定的功能块访问。

应首先生成功能块，然后生成它的背景数据块。在生成背景数据块时指明它的类型为背景数

据块（Instance），并指明功能块的编号。在调用功能块时使用不同的背景数据块，可以控制多个同类的对象。例如一个用于电动机控制的 FB，可以通过对每个不同的电动机使用不同的背景数据集，来控制多台电动机（见图 6-1）。

（3）系统功能块（SFB）和系统功能（SFC）

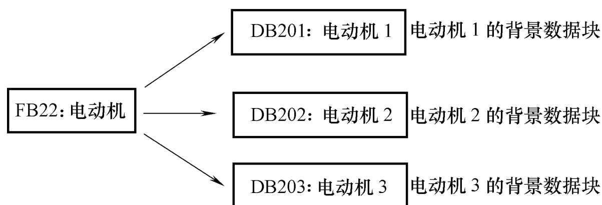


图 6-1　用于不同对象的背景数据块

系统功能块（SFB）和系统功能（SFC）是 S7 CPU 提供的标准的已经为用户编制好程序的块，用户可以直接调用它们，以便高效地编制自己的程序，但不能修改这些功能块。它们是操作系统固有的一部分，不占用户程序空间。其中系统功能块（SFB）有存储功能，其变量保存在指定给它的背景数据块中。

（4）系统数据块（SDB）

系统数据块（SDB）是由 STEP7 产生的程序存储区，包含系统组态数据，例如硬件模块参数和通信连接参数等用于 CPU 操作系统的数据。

（5）块的调用

在程序编制过程中，可以用 CALL、CU（无条件调用）和 CC（RLO = 1 时调用）指令调用没有参数的 FC 和 FB。这里需要注意用 CALL 指令调用 FB 和 SFB 时，必须指定背景数据块，而且静态变量和临时变量不能出现在调用指令中。

二、用户程序使用的堆栈

堆栈（见图 6-2）是 CPU 中的一块特殊存储区，它采用“先入后出”的规则存入和取出数据。堆栈最上面的存储单元称为栈顶，要保存的数据从栈顶压入堆栈时，栈中原有的数据依次向下移动一个位置，最下面一个存储单元的数据丢失。同理，在取出栈顶的一个数据后，栈中所有的数据依次向上移动一个位置。堆栈的这种“先入后出”的存取规则刚好满足块的调用要求，因此在程序设计中得到了普遍的应用。

下面介绍 STEP7 中 3 种不同的堆栈。

1. 局域数据堆栈（L）

局域数据堆栈用来存储块的局域数据区的临时变量、组织块的起动信息、块传递参数的信息和梯形图程序的中间结果，局域数据可以按位、字节、字和双字来存取，例如 L0.0、LB9、LW4 和 LD52。

各逻辑块均有自己的局域变量表，局域变量仅在它被创建的逻辑块中有效。对组织块编程时，可以声明临时变量（TEMP）。临时变量仅在块被执行的时候使用，块执行完后将被别的数据覆盖。

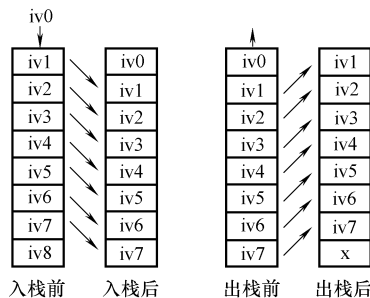


图 6-2　堆栈操作

在首次访问局域数据堆栈时，应对局域数据初始化。每个组织块需要 20B 的局域数据来存储它的起动信息。

CPU 分配给当前正在处理的块的临时变量（即局域数据）的存储器容量是有限的，这一存储区（即局域堆栈）的大小与 CPU 的型号有关。CPU 给每一优先级分配了相同数量的局域数据区，这样可以保证不同优先级的 OB 都有它们可以使用的局域数据空间。

图 6-3 中的 FB1 调用功能 FC2，FC2 的执行被组织块 OB81 中断。图 6-3 中给出了局域数据

堆栈中局域数据的存放情况。

在局域数据堆栈中，并非所有的优先级都需要相同数量的存储区。通过在 STEP7 设置参数，可以给 S7-400 CPU 和 CPU 318 的每一优先级指定不同大小的局域数据区。其余的 S7-300 CPU 每一优先级的局域数据区的大小是固定的。

2. 块堆栈（B 堆栈）

如果一个块的处理因为调用另外一个块，或者被更高优先级的块中止，或者被错误的服务中止，CPU 将在块堆栈中存储以下信息：

1）被中断的块的类型（OB、FB 、FC 、SFB 、SFC）、编号、优先级和返回地址。

2）从 DB 和 DI 寄存器中获得的块被中断时，打开的共享数据块和背景数据块的编号（即块存储器 DB、DI 被中断前的内容）。

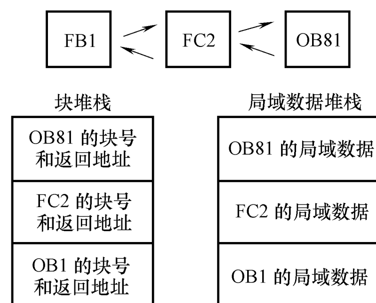


图 6-3 块堆栈与局域数据堆栈

3）局域数据堆栈的指针（被中断块的 L 堆栈地址）。

利用这些数据，可以在中断它的任务处理完后恢复被中断的块的处理。在多重调用时，堆栈可以保存参与嵌套调用的几个块的信息。

CPU 处于 STOP 模式时，可以在 STEP7 中显示 B 堆栈中保存的在进入 STOP 模式时各种处理完的块，在 B 堆栈中，块按照它们被处理的顺序排列。

STEP7 中可使用的 B 堆栈的大小是有限的，这与 CPU 的型号有关。

3. 中断堆栈（I 堆栈）

如果程序的执行被优先级更高的 OB 中断，操作系统将保存下述寄存器的内容：当前累加器和地址寄存器的内容、数据块寄存器 DB 和 DI 的内容、局域数据的指针、状态字、MCR（主控继电器）寄存器和 B 堆栈的指针。

新的 OB 执行完后，操作系统从中断堆栈中读取信息，从被中断块的被中断的地方开始继续执行程序。

CPU 在 STOP 模式时，可以在 STEP7 中显示 I 堆栈中保存的数据，用户可以由此找出使 CPU 进入 STOP 模式的原因。

三、STEP7 编程方式

STEP 7 提供了 3 种编程方法供用户选用，它们分别是线性化编程、模块化编程和结构化编程。

1. 线性化编程

线性化编程是指将整个用户程序写在一个指令连续的块中，处理器循环扫描时不断地依次执行块中的每条指令。这种方式的程序结构简单，不涉及功能块、功能、数据块、局域变量和中断等比较复杂的概念，适合比较简单的控制任务。

由于所有的指令都在一个块中，即使程序中的某些部分在大多数时候并不需要执行，然而在每个扫描周期都要执行所有的指令，因此线性化编程方法不能有效地利用 CPU。

2. 模块化编程

模块化编程是将用户程序分成相对独立的指令块，每个块包含完成某些任务的逻辑指令。各备用块的执行顺序由组织块 OB1（即主程序）中的指令决定。功能和功能块（即子程序）用来完成不同的过程任务。被调用的块执行完后，返回到 OBI 中程序块的调用点，继续执行 OB1。

与线性化编程方法相比，由于只是在需要时才调用有关的程序块，提高了 CPU 的利用效率。

3. 结构化编程

结构化编程要求用户程序提供一些通用的指令块，以便控制一类相似或相同的部件，从而将复杂的自动化任务分解为能够反映过程的工艺、功能或可以反复使用的小任务，这些任务由相应的程序块（或称逻辑块）来表示，程序运行时所需的大量数据和变量存储在数据块中。某些程序块可以用来实现相同或相似的功能。这些程序块是相对独立的，它们被 OB1 或别的程序块调用。

结构化编程方法适合复杂的控制任务，并支持多人协同编写大型用户程序。具有程序结构层次清楚、部分程序标准化、易于修改、程序调试简单等优点。

第二节　功能块与功能的调用

功能和功能块由两个主要部分组成：一部分是每个功能或功能块的变量声明表，变量声明表声明此块的局域数据；另一部分是逻辑指令组成的程序，程序要用到变量声明表中给出的局域数据。

当调用功能块或功能时，需提供块执行时要用到的数据或变量，也就是将外部数据传递给功能块或功能，这就是参数传递。参数传递的方式使得功能块具有通用性，它可能被其他功能块调用，以完成多个类似的控制任务。

一、局域变量的类型

局域变量类型见表 6-2。

表 6-2　局域变量类型

变量名	类型	说　明
输入参数	IN	由调用它的块提供的输入参数
输出参数	OUT	返回给调用它的块的输出参数
I/O 参数	IN_OUT	参数的值由调用它的块提供,由逻辑块修改后返回给调用它的块
静态变量	TEMP	静态变量存储在背景数据块中,块调用结束后,其内容被保留
临时变量	STAT	临时变量暂时保存在 L 堆栈中,块执行结束后,其内容被覆盖掉

在变量声明表中赋值时，不需要指定存储器地址；根据各变量的数据类型，程序编辑器自动地为所有局域变量指定存储器地址。表 6-3 给出了全局变量与局部变量的使用情况。

表 6-3　全局变量与局部变量

全局变量(在整个程序中使用)	局域变量(只能在一个块中使用)	
PII/PIQ、I/O、M/T/C、DB 区	临时变量:临时存储于 L 堆栈中,可以用于 FB、FC、OB 中,对应块执行完后被删除	静态变量:只能在 FB 中使用,对应块执行完后永久保留在背景数据块中

在变量声明表中选中 ARRAY（数组）时，用鼠标点击相应行的地址单元。如果想要选中一个结构（Structure），用鼠标选中结构的第一行或最后一行的地址单元，即有关键字 STRUCT 或 END_STRUCT 的那一行。若要选中结构中的某一参数，用鼠标点击该行的地址单元即可。

二、功能块与功能的调用

CPU 提供堆栈（B 堆栈）来存储与处理被中断块的有关信息。当发生块调用或有来自更高优先级的中断时，就有相关的信息存储在 B 堆栈里，并影响部分内存和存储器。图 6-4 显示了调用指令对 CPU 内存的影响。

1. 调用功能块（FB）

当调用功能块（FB）时，将会发生以下事件：

1）调用块的地址和返回位置存储在块堆栈中，调用块的临时变量压入 L 堆栈。

2）数据块（DB）寄存器内容与 DI 寄存器内容交换。

3）新的数据块地址装入 DI 寄存器。

4）被调用块的实参装入 DB 和 L 堆栈上部。

5）当功能块（FB）结束时，先前块的现场信息从块堆栈弹出，临时变量弹出 L 堆栈。

6）DB 和 DI 寄存器内容交换。

2. 调用功能（FC）

当调用功能（FC）时将有以下事件发生：

1）功能（FC）实参的指针被存储到调用块的 L 堆栈中。

2）调用块的地址和返回位置存储在块堆栈中，调用块的临时变量压入 L 堆栈。

3）功能（FC）存储临时变量的 L 堆栈区被推到堆栈上部。

4）当功能（FC）结束时，先前块的现场信息存储在块堆栈中，临时变量弹出 L 堆栈。

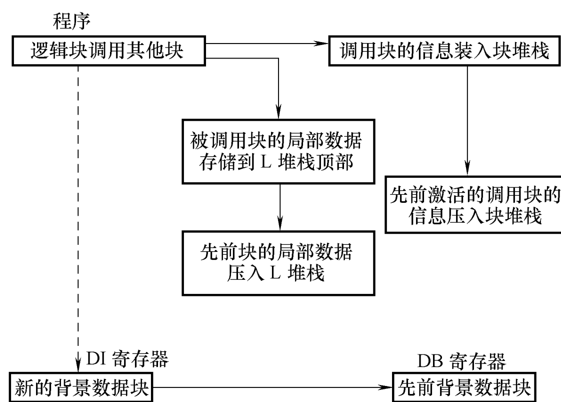


图 6-4　调用指令对 CPU 内存的影响

因为功能（FC）不用背景数据块，不能分配初始数值给功能（FC）的局域数据，所以必须给功能（FC）提供实参。

下面以发动机控制系统的用户程序为例，介绍生成和调用功能块和功能的方法。

（1）创建项目

生成一个新项目最简单的方法是使用“NEW PROJECT”向导，具体方法是在计算机的“桌面”上双击“SIMATIC Manager ”图标，在弹出的新项目向导中点击“NEXT”按钮，依次选择 CPU 的型号、MPI 站地址、需要编程的组织块和使用的编程语言等，最后设置项目的名称为“发动机控制”。

（2）生成用户程序结构

图 6-5 中的组织块 OB1 是主程序，用一个名为“发动机控制”的功能块 FB1 来分别控制汽油机和柴油机，控制参数在背景数据块 DB1 和 DB2 中。控制汽油机时调用 FB1 和名为“汽油机数据”的背景数据块 DB1，控制柴油机时调用 FB1 和名为“柴油机数据”的背景数据块 DB2。此外，控制汽油机和柴油机时还用不同的实参分别调用名为“风扇控制”的功能 FC1。图 6-6 是程序设计好后 SIMATIC 管理器中的块。

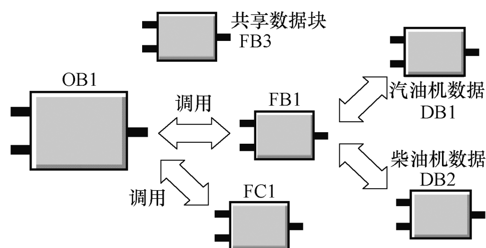


图 6-5　程序结构

（3）编制符号表与变量声明表

1）符号表：为了使程序易于理解，可以给变量指定符号。表 6-4 是发动机控制项目的符号表，符号表中定义的变量是全局变量，可供所有的逻辑块使用。

2）变量声明表：表 6-5 列出了发动机控制例程中 FB1 的局域变量。表中 Bool 变量的初值 FALSE 即二进制 0。预置转速是固定值，在变量声明表中作为静态参数被存储，称为“静态局域变量”。



图 6-6 SIMATIC 管理器中的块

表 6-4 符号表

符号	地址	符号	地址	符号	地址	符号	地址
汽油机数据	DB1	起动汽油机	I1. 0	柴油机转速	MW4	柴油机达设定转速	Q5. 5
柴油机数据	DB2	关闭汽油机	I1. 1	主程序	OB1	柴油机风扇运行	Q5. 6
共享数据	DB3	汽油机故障	I1. 2	自动模式	Q4. 2	汽油机风扇运行	T1
发动机控制	FB1	起动柴油机	I1. 3	汽油机运行	Q5. 0	柴油机风扇延时	T2
风扇控制	FC1	关闭柴油机	I1. 4	汽油机达设定转速	Q5. 1		
自动按钮	I0. 5	柴油机故障	I1. 5	汽油机风扇运行	Q5. 2		
手动按钮	I0. 6	汽油机转速	I1. 6	柴油机运行	Q5. 4		

表 6-5 FB1 的变量声明表

名称	数据类型	地址	声明	初值	说明
Switch_On	Bool	0. 0	IN	FALSE	起动按钮
Switch_Off	Bool	0. 1	IN	FALSE	停车按钮
Failure	Bool	0. 2	IN	FALSE	故障信号
Actual_Speed	Int	2. 0	IN	0	实际转速
Engine_On	Bool	4. 0	OUT	FALSE	发动机输出信号
Preset_Speed_Reached	Bool	4. 1	OUT	FALSE	达到预置转速
Preset_Speed	Int	6. 0	STAT	1500	预置转速

如果控制功能不需要保存它自己的数据，也可以用功能（FC）来编程。与功能块（FB）相比较，FC 不需要配套的背景数据块。

在功能的变量声明表中可以使用的参数类型有 IN、OUT、IN_OUT、TEMP 和 RETURN（返回参数），功能不能使用静态（STAT）局域数据。

表 6-6 是 FC1 中使用的变量。在变量声明表中不能用汉字作变量的名称。

表 6-6 FC1 的变量声明表

名称	数据类型	声明	说 明
Engine_On	Bool	IN	输入信号，发动机起动
Timer_Function	Timer	IN	停机延时的定时器功能
Fan_On	Bool	OUT	用于控制风扇的输出信号

FC1 用来控制发动机的风扇，要求在起动发动机的同时起动风扇，发动机停车后，风扇继续运行 4s 后停转，因此使用了延时断开定时器（S_OFFDT）。图 6-7 是 FC1 的梯形图。

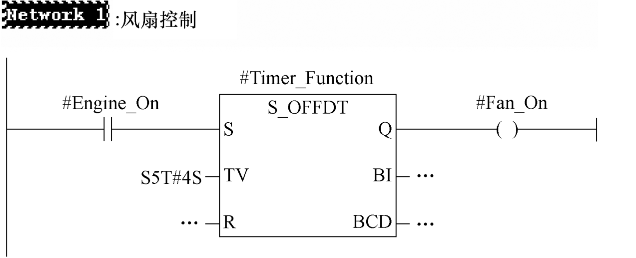


图 6-7 FC1 的梯形图

（4）编制程序

程序可以用语句表或梯形图两种形式来编制，下面分别给出了这两种程

序的编制方法。

1）梯形图主程序 OB1 如图 6-8 所示。

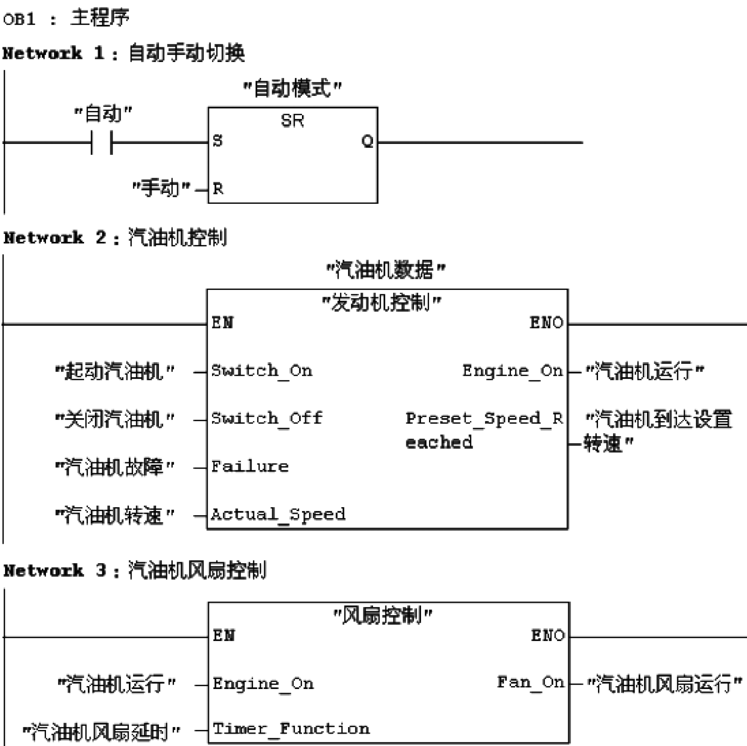


图 6-8　梯形图主程序 OB1

2）语句表程序如下：

```
Network1：自动手动切换
A          "自动"
S          "自动模式"
A          "手动"
R          "自动模式"
Network2：汽油机控制
CALL  "发动机控制","汽油机数据"
Switch_On              ：= "起动汽油机"
Switch_Off             ：= "关闭汽油机"
Failure                ：= "汽油机故障"
Actual_Speed           ：= "汽油机转速"
Engine_On              ：= "汽油机运行"
Preset_Speed_Reached   ：= "汽油机到达设置转速"
Network3：汽油机风扇控制
CALL   "风扇控制"
Engine_On              ：= "汽油机运行"
Timer_Function         ：= "汽油机风扇延时"
```

Fan_On : =“汽油机风扇运行”

在 OB1 中，用 CALL 指令调用功能块 FB1。方框内的“发动机控制”是功能块 FB1 的符号名，方框上面的“汽油机数据”是对应的背景数据块 DB1 的符号名。方框内是功能块的形参，方框外是对应的实参。方框的左边是块的输入量，右边是块的输出量。功能块的符号名是在符号表中定义的。

两次调用功能块“发动机控制”时，功能块的输入变量和输出变量不同，除此之外，分别使用汽油机的背景数据块“汽油机数据”和柴油机的背景数据块“柴油机数据”。两个背景数据块中的变量相同，区别仅在于变量的实际参数（即实参）不同和静态参数（例如预置转速）的初值不同。背景数据块中的变量与“发动机控制”功能块的变量声明表中的变量相同（不包括临时变量 TEMP）。

第三节 数 据 块

一、数据块的生成与使用

数据块用来分类存储设备或生产线中变量的值，数据块也是用来实现各逻辑块之间的数据交换、数据传递和共享数据的重要途径。与逻辑块不同，数据块只有变量声明部分，没有程序指令部分。

1. 数据块的类型

数据块分为共享数据块（DB）和背景数据块（DI）两种。

共享数据块又称为全局数据块，它不附属于任何逻辑块。在共享数据块中和全局符号表中声明的变量都是全局变量。用户程序中所有的逻辑块（FB、FC、OB 等）都可以使用共享数据块和全局符号表中的数据。

背景数据块是专门指定给某个功能块（FB）或系统功能块（SFB）使用的数据块，它是 FB 或 SFB 运行时的工作存储区。当用户将数据块与某一功能块相连时，该数据块即成为该功能块的背景数据块，功能块的变量声明表决定了它的背景数据块的结构和变量。不能直接修改背景数据块，只能通过对应的功能块的变量声明表来修改它。调用 FB 时，必须同时指定一个对应的背景数据块。只有 FB 才能访问存放在它的背景数据块中的数据。

在符号表中，共享数据块的数据类型是它本身，背景数据块的数据类型是对应的 FB。

多次使用同一 FB 时，需要调用不同的背景数据块，具体做法是将这些数据块中的数据存放在一个多重背景数据块中，但需要增加一个管理多重背景的 FB。

2. 定义数据块

在编程阶段和程序运行中都能定义数据块，大多数数据块是在编程阶段用 STEP 7 开发软件包定义的，定义内容包括数据块号和块中的变量。定义完成后，数据块中变量的顺序及类型决定了数据块的数据结构，变量的数量决定了数据块的大小。数据块在使用前，必须作为用户程序的一部分下载到 CPU 中。

3. 生成共享数据块

在 SIMATIC 管理器中用鼠标右键点击 SIMATIC 管理器的块工作区，在弹出的菜单中选择“Insert New Object”→“Data Block”命令，就可以生成新的数据块。

数据块有两种显示方式，即声明表显示方式和数据显示方式，菜单命令“View”→“Declaration View”和“View”→“Data View”分别指定声明表显示方式和数据显示方式。声明表显示状态用于定义和修改共享数据块中的变量，指定它们的名称、类型和初值，STEP7 根据数据类型给

出默认的初值，用户可以修改初值。可以用中文给每个变量加上注释，声明表中的名称只能使用字母、数字和下划线，地址是 CPU 自动指定的。在数据显示状态，显示声明表中的全部信息和变量的实际值，用户只能改变每个元素的实际值。复杂数据类型变量的元素（例如数组中的各元素）用全名列出。如果用户输入的实际值与变量的数据类型不符，STEP7 将用红色显示错误的数据。在数据显示状态下，用菜单命令“Edit”→“Inicialize Data Block”可以恢复变量的初始值。

4. 生成背景数据块

要生成背景数据块，首先应生成对应的 FB，然后再生成背景数据块。在 SIMATIC 管理器中，用菜单命令“Insert”→“S7 Block”→“Data Block”生成数据块，在弹出的窗口中，选择数据块的类型为背景数据块（Instance），并输入对应的 FB 的名称。操作系统在编译 FB 时将自动生成 FB 对应的背景数据块中的数据，其变量与对应的 FB 的变量声明表中的变量相同，不能在背景数据块中增减变量，只能在数据显示（Data View）方式下修改其实际值。在数据块编辑器的“View”菜单中选择是声明表显示方式还是数据显示方式。

5. 访问数据块

在访问数据块时，需要指明被访问的是哪一个数据块，以及访问该数据块中的哪一个数据。有两种访问数据块中的数据的方法。

1）访问数据块中的数据时，需要用 OPN 指令先打开它。

2）在指令中同时给出数据块的编号和数据在数据块中的地址，直接访问数据块中的数据。例如 DBZ. DBX2. 0，其中 DBZ 是数据块的名称，DBX2. 0 是数据块内第 2 个字节的第 0 位。这种访问方法不容易出错，建议尽量使用这种方法。

二、数据块中的数据类型

1. 基本数据类型

数据块中基本数据类型包括位（Bool）、字节（Byte）、字（Word）、双字（Dword）、整数（INT）、双整数（DINT）和浮点数（Float，或称实数 Real）等。

2. 复合数据类型

复合数据类型包括日期和时间（DATE_AND_TIME）、字符串（STRING）、数组（ARRAY）、结构（STRUCT）和用户定义数据类型（UDT）。其中，日期和时间用 8 个字节的 BCD 码来存储。第 0 ~ 5 号字节分别存储年、月、日、时、分和秒，毫秒存储在字节 6 和字节 7 的高 4 位，星期存放在字节 7 的低 4 位。例如 2004 年 7 月 27 日 12 点 30 分 25. 123 秒可以表示为 DT#04-07-27-12：30：25. 123。

字符串（STRING）由最多 254 个字符（CHAR）和 2 字节的头部组成。字符串的默认长度为 254，通过定义字符串的长度可以减少它占用的存储空间。

3. 数组

数组（ARRAY）是同一类型的数据组合而成的一个单元。生成数组时，应指定数组的名称，声明数组的类型时要使用关键字 ARRAY，用下标指定数组的维数和大小，数组的维数最多为 6 维。例如图 6-9 给出了一个二维数组 PRESS［1. . 2，1 . . 3］，共有 6 个整数元素，图中的每一小格为二进制的 1 位，每个元素占两行（两个字节）。方括号中的数字用来定义每一维的起始元素和结束元素在该维中的编号，可以取 –32768 ~ 32767 之间的整数。各维之间的数字用逗号隔开，每一维开始和结束的编号用两个小数点隔开，如果某一维有 n 个元素，该维的起始元素和结束元素的编号一般采用 1 和 n，例如 PRESS［ 1. . 2，1 . . 3］，第一个整数是 PRESS［1，1］，第三个整数是 PRESS［1，3］，第四个整数是 PRESS［2，1］，第六个整数是 PRESS［2，3］。

访问数组中的数据时，需要指出数据块和数组的名称，以及数组元素的下标，例如，如果数

组 ARRAY 是数据块 TANK 的一部分，则访问格式为“TANK“. PRESS［2，1］。其中，TANK 是数据块的符号名，PRESS 是数组的名称，它们用英语的句号分开。方括号中是数组元素的下标，该元素是数组中的第 4 个元素（见图 6-9）。

如果在块的变量声明表中声明形参的类型为 ARRAY，可以将整个数组而不是某些元素作为参数来传递。在调用块时也可以将某个数组元素赋值给同一类型的参数。将数组作为参数传递时，要求形参和实参必须有相同的数据组织结构、相同的数据类型，并按相同的数据排列。

4. 结构

结构（STRUCT）是不同类型的数据的组合。可以用基本数据类型、复杂数据类型（包括数组和结构）和用户定义数据类型作为结构中的元素，例如一个结构由数组和结构组成，结构可以嵌套 8 层。用户可以把过程控制中有关的数据统一组织在一个结构中，作为一个数据单元来使用，而不是使用大量的单个的元素，这一点为统一处理不同类型的数据或参数提供了方便。

与数组一样，结构既可以在数据块中定义，也可以在逻辑块的变量声明表中定义。可以为结构中各个元素设置初值（Initial Value）并加上注释（Comment），可以用结构中的元素的绝对地址或符号地址来访问结构中的元素。例如数据块 TANK 内结构 STACK 的元素 AMOUNT，用符号地址访问时应表示为“TANK”. SIACK. AMOUNT。再如 STACK 从 DB1 的字节 12 开始存放，用绝对地址来访问时应表示为 DB1. DBW12。

1,1 整数
1,2 整数
1,3 整数
2,1 整数
2,2 整数
2,3 整数

图 6-9　二维数组 PRESS［1..2，1..3］的结构

如果在块的变量声明表中，声明形参的类型为 STRUCT，可以将整个结构而不是某些元素作为参数来传递。在调用块时也可以将某个结构元素赋值给同一类型的参数。另外，将结构作为参数传递时，作为形参和实参的两个结构必须有相同的数据结构，即相同数据类型的结构元素和相同的排列顺序。

5. 用户定义数据类型

STEP 7 允许用户将基本数据类型或复合数据类型组合成自定义数据类型（UDT）。它是一种特殊的数据结构，定义好后可以在用户程序中多次使用。

使用用户定义数据类型时，只需要对它定义一次，就可以用它来产生大量的具有相同数据结构的数据块，可以用这些数据块来输入用于不同目的的实际数据。例如可以生成用于颜料混合配方的 UDT，然后用它生成用于不同颜色配方的数据组合。

第四节　多重背景

在用户程序中使用多重背景可以减少背景数据块的数量。以发动机控制程序为例，原来用 FB1 控制汽油机和柴油机时，分别使用了背景数据块 DB1 和 DB2。使用多重背景时只需要一个背景数据块（DB10），但是需要增加一个功能块 FB10 来调用作为“局域背景”的 FB1，FB1 的数据存储在 FB10 的背景数据块 DB10 中，DB10 是自动生成的。不需要给 FB1 分配背景数据块，即原来的 DB1 和 DB2 被 DB10 代替，但是需要在 FB10 的变量声明表中声明静态局域数据（STAT）FB1。多重背景的程序结构如图 6-10 所示。

一、多重背景功能块的生成

生成多重背景功能块 FB10 时，首先应激活“Multiple Instance FB”（多重背景功能块）选项。生成多重背景功能块 FB10 时，首先需要生成 FB1。为调用 FB1，在 FB10 的变量声明表中

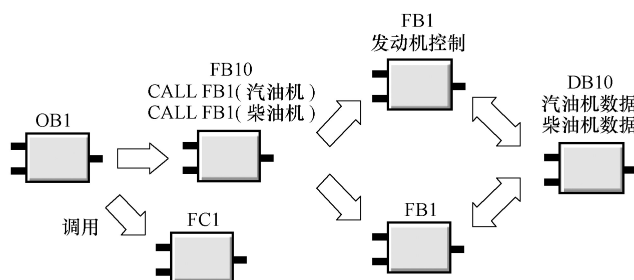


图 6-10　多重背景的程序结构

（见图 6-11）声明了两个名为“Petrol_Engine（汽油机）”和“Diesel_Engine（柴油机）”的静态变量（STAT），其数据类型为 FB1。生成 FB10 后，“Petrol_Engine”和“Diesel_Engine”将出现在管理器编程元件目录的“Multiple Instances（多重背景）”文件夹内。可以将它们“拖放”到 FB10 中，然后指定它们的输入参数和输出参数。

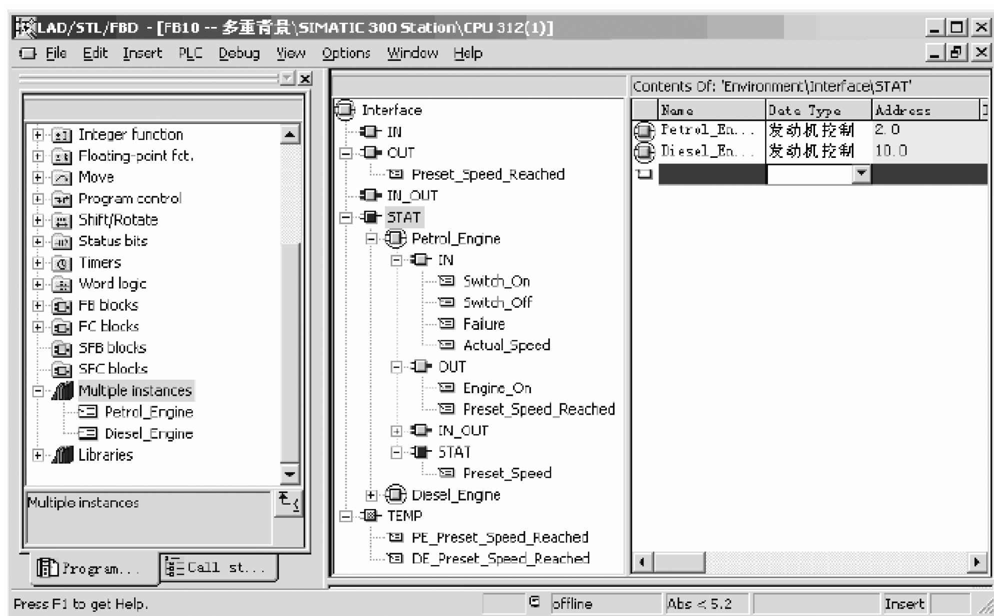


图 6-11　FB10 的变量声明表

二、多重背景功能块的编程

多重背景功能块的编程方法包括梯形图编程、功能块编程和语句表编程 3 种。

1. 用梯形图编程

图 6-12 是 FB10 中的梯形图程序。

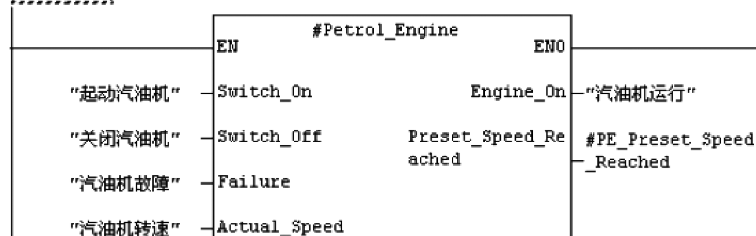
汽油机和柴油机的数据均存储在多重背景数据块 DB10 中，该数据块代替了原有的背景数据块 DB1 和 DB2。生成 DB10 时，应将它设计成背景数据块，对应的功能块为 FB10，DB10 中的变量是自动生成的，与 FB10 的变量声明表中的相同。打开 DB10，执行菜单命令“View”→“Data View”，可以修改预置转速的实际值。

2. 用功能块图编程

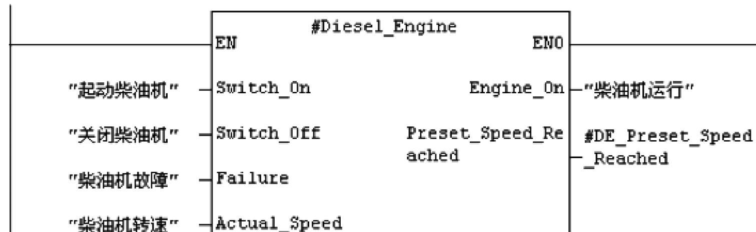
图 6-13 是 FB10 中的功能块图程序。

FB10 ： 多重背景举例

Network 1： 汽油机控制



Network 2 : 柴油机控制



Network 3 : 两台发动机都达到预置转速

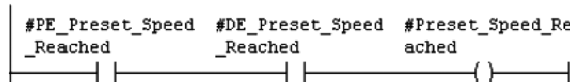
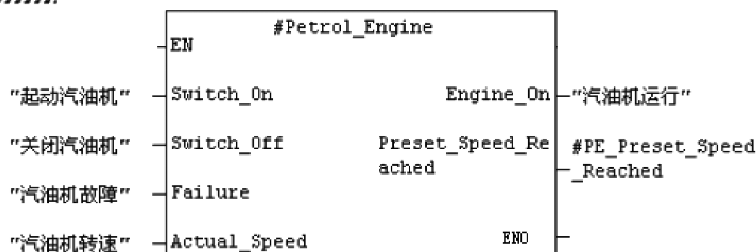


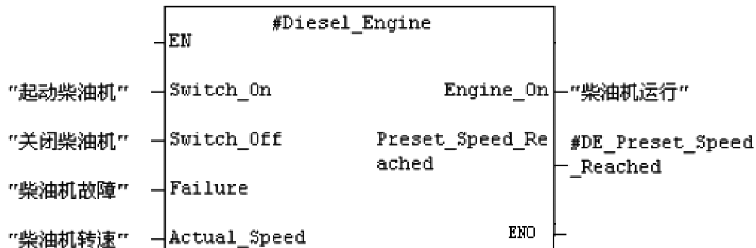
图 6-12　多重背景功能块 FB10 的梯形图程序

FB10 ： 多重背景举例

Network 1： 汽油机控制



Network 2 : 柴油机控制



Network 3 : 两台发动机都达到预置转速

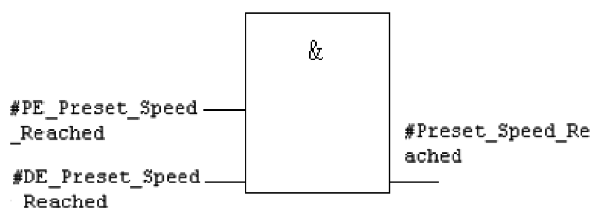


图 6-13　FB10 中的功能块图程序

3. 用语句表编程

用语句表编写的 FB10 的程序如下。

```
Network1：汽油机控制
CALL    “发动机控制”，“汽油机数据”
Switch_On              ：＝“起动汽油机”
Switch_Off             ：＝“关闭汽油机”
Failure                ：＝“汽油机故障”
Actual_Speed           ：＝“汽油机转速”
Engine_On              ：＝“汽油机运行”
Preset_Speed_Reached   ：＝“汽油机到达设置转速”
Network2：柴油机控制
CALL    “发动机控制”，“柴油机数据”
Switch_On              ：＝“起动柴油机”
Switch_Off             ：＝“关闭柴油机”
Failure                ：＝“柴油机故障”
Actual_Speed           ：＝“柴油机转速”
Engine_On              ：＝“柴油机运行”
Preset_Speed_Reached   ：＝“柴油机到达设置转速”
Network3：两台发动机都达到预置转速
A          “汽油机到达设置转速”
A          “柴油机到达设置转速”
=          “汽油机柴油机都到达设置转速”
```

三、在 OB1 中调用多重背景

给 OB1 编程之前，应先打开符号表，输入 FB10 和 DB10 的符号名，然后保存退出。前面的“发动机控制”项目中 OB1 对 FB1 的两次调用，被 OB1 对 FB10（符号名为“发动机”）的调用代替，调用时还指定了背景数据块 DB10（符号名为“多重背景数据块”）。FB10 的输出信号“Preset_Speed _Reached”送给符号名为“两台达到设置转速”的共享数据 Q5. 1。OB1 中调用多重背景如图 6-14 所示。

```
Network4：调用多重背景
CALL    “发动机”，“多重背景数据块”
Preset_Speed_Reached ：＝“两台都达到设置转速”
```

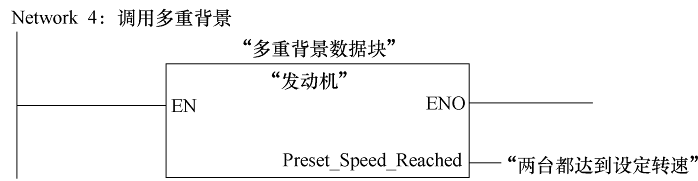


图 6-14　OB1 中调用多重背景

使用多重背景时应注意以下问题：

1）首先应生成需要多次调用的功能块（例如上例中的 FB1）。

2）管理多重背景的功能块（例如上例中的 FB10）必须设置为有多重背景功能。

3）在管理多重背景的功能块的变量声明表中，为被调用的功能块的每一次调用定义一个静态（STAT）变量，以被调用的功能块的名称（例如 FB1）作为静态变量的数据类型。

4）必须有一个背景数据块（例如上例中的 DB10）分配给管理多重背景的功能块。背景数据块中的数据是自动生成的。

5）多重背景只能声明为静态变量（声明类型为“Stat”）。

第五节　组织块与中断处理

组织块（OB）是操作系统与用户程序之间的接口。S7 提供了各种不同的组织块，用组织块可以创建在特定的时间执行的程序和响应特定事件的程序，例如延时中断 OB、外部硬件中断 OB 和错误处理 OB 等。

一、中断的基本概念

1. 中断过程

中断处理用来实现对特殊内部事件或外部事件的快速响应。如果没有中断，CPU 循环执行组织块 OB1。当 CPU 检测到中断源的中断请求时，操作系统在执行完当前程序的当前指令（即断点处）后，立即响应中断。CPU 暂停正在执行的程序，调用中断源对应的中断程序。在 S7-300/400 中，中断用 OB 来处理。执行完中断程序后，返回被中断的程序的断点处继续执行原来的程序。

PLC 的中断源可能来自 I/O 模块的硬件中断，或是 CPU 模块内部的软件中断，例如日期时间中断、延时中断、循环中断和编程错误引起的中断。

如果在执行中断程序（OB）时，又检测到一个中断请求，CPU 将比较两个中断源的中断优先级。如果优先级相同，按照产生中断请求的先后次序进行处理。如果后者的优先级比正在执行的 OB 的优先级高，将中止当前正在处理的 OB，改为调用较高优先级的 OB。这种处理方式称为中断程序的嵌套调用。

一个 OB 被另一个 OB 调用时，操作系统对现场进行保护。被中断的 OB 的局域数据压入 L 堆栈（局域数据堆栈），被中断的断点处的现场信息保存在 I 堆栈（中断堆栈）和 B 堆栈（块堆栈）中。

中断程序不是由程序块调用，而是在中断事件发生时由操作系统调用。因为不能预知系统何时调用中断程序，中断程序不能改写其他程序中可能正在使用的存储器，应在中断程序中尽可能地使用局域变量。

编写中断程序时，应使中断程序尽量短小，以减少中断程序的执行时间，减少对其他处理的延迟，否则可能引起主程序控制的设备操作异常。设计中断程序时应遵循“越短越好”的原则。

2. OB 的分类

OB 只能由操作系统起动，它由变量声明表和用户编写的控制程序组成。

（1）起动 OB

起动 OB 用于系统初始化，CPU 上电或操作模式改为 RUN 时，根据起动的方式执行起动程序 OB100 ~ OB102 中的一个。

（2）循环执行的 OB

需要连续执行的程序存放在 OBI 中，执行完后又开始新的循环。

（3）定期执行的 OB

包括日期时间中断组织块 OB10 ~ OB17 和循环中断组织块 OB30 ~ OB38，可以根据设定的日

期时间或时间间隔执行中断程序。

（4）事件驱动的 OB

延时中断 OBZO ~ OB23 在过程事件出现后延时一定的时间再执行中断程序；硬件中断OB40 ~ OB47 用于需要快速响应的过程事件，事件出现时马上中止循环程序，执行对应的中断程序。异步错误中断 OB80 ~ OB87 和同步错误中断 OB121、OB122 用来决定在出现错误时系统如何响应。

3. 中断的优先级

中断的优先级也就是 OB 的优先级，较高优先级的 OB 可以中断较低优先级的 OB 的处理过程。如果同时产生的中断请求不止一个，最先执行优先级最高的 OB，然后按照优先级由高到低的顺序执行其他 OB。

中断的优先级由低到高的排列顺序是：背景循环、主程序扫描循环、日期时间中断、时间延时中断、循环中断、硬件中断、多处理器中断、冗余错误、异步故障（OB80 ~ 87）、起动和 CPU 冗余。

需要指出的是，同一个优先级可以分配给几个 OB，具有相同优先级的 OB 按起动它们的事件出现的先后顺序处理。被同步错误起动的故障 OB 的优先级与错误出现时正在执行的 OB 的优先级相同。

4. 对中断的控制

日期时间中断和延时中断有专用的允许处理中断和禁止中断的系统功能（SFC）。其中，SFC39“DIS_INT”用来禁止所有的中断、某些优先级范围的中断或指定的某个中断；SFC 40“EN_INT”用来激活（使能）新的中断和异步错误处理。如果用户希望忽略中断，可以下载一个只有块结束指令 BEU 的空的 OB；SFC41“DIS_AIRT”延迟处理比当前优先级高的中断和异步错误；SFC 42“EN_ AIRT”允许立即处理被 SFC 41 暂时禁止的中断和异步错误。

二、组织块的变量声明表

OB 由操作系统调用，OB 没有背景数据块，也不能为自己声明静态变量，因此 OB 的变量声明表中只有临时变量，其临时变量可以是基本数据类型、复合数据类型或数据类型 ANY。

操作系统为所有的 OB 声明了一个包含 OB 的起动信息的变量声明表，声明表中变量的具体内容与组织块的类型有关。用户可以通过 OB 的变量声明表获得与起动 OB 的原因有关的信息。OB 的变量声明表见表 6-7。

表 6-7　OB 的变量声明表

字节地址	内　　容	字节地址	内　　容
0	事件级别与标识符	3	OB 号，例如 OB40 的块号为 40
1	用代码表示与起动 OB 的事件有关的信息	4 ~ 11	其他附加信息
2	优先级，例如 OB40 的优先级为 16	12 ~ 19	OB 被起动的日期和时间（年、月、时、分、秒、毫秒、星期）

三、日期时间中断组织块（OB10 ~ OB17）

S7 CPU 提供日期时间中断 OB，这些 OB 在特定的日期和时间或以一定的间隔由操作系统调用执行。CPU 可以使用的日期时间中断 OB 的个数与 CPU 的型号有关。例如 S7-300 只能用 OB10。

1. 设置和起动日期时间中断

为了起动日期时间中断，用户首先必须设置日期时间中断的参数，然后再激活它。有以下 3 种方法可以起动日期时间中断。

1）在用户程序中用 SFC 28“SET_TINT”和 SFC 30“ACT_TINT”设置并激活日期时间

中断。

2）在硬件组态工具中设置和激活。其具体步骤是：在 STEP7 中打开硬件组态工具，双击机架中 CPU 模块所在的行，打开设置 CPU 属性的对话框，点击“Time-Of-Day Interrupts”选项卡，设置起动时间日期中断的日期和时间，选中“Active”（激活）多选框，在“Execution”列表框中选择执行方式。将硬件组态数据下载到 CPU 中，就可以实现日期时间中断的自动起动。

3）在用户程序中用 SFC 30“ACT_TINT”激活日期时间中断。

2. 查询日期时间中断

要想查询设置了哪些日期时间中断，以及这些中断什么时间发生，用户可以调用“QRY_TINT”，或查询系统状态表中的“中断状态”表。SFC31 输出的状态字节 STATUS 见表 6-8 。

表 6-8　SFC 31 输出的状态字节 STATUS

位	取值	意　义	位	取值	意　义
0	0	日期时间中断已被激活	4	0	没有装载日期时间中断 OB
1	0	允许新的日期时间中断	5	0	日期时间中断 OB 的执行没有被激活的测试功能禁止
2	0	日期和时间中断未被激活或时间已过去	6	0	以基准时间为日期时间中断的基准
3	0	—	7	1	以本地时间为日期时间中断的基准

3. 终止与激活日期时间中断

用 SFC 29“CAN_TINT”，用户可以取消那些还没有执行的日期时间中断。用 SFC 28“SET_TINT”可以重新设置那些被禁止的日期时间中断，用 SFC 30“ACT_TINT”重新激活日期时间中断。

四、时间延时中断组织块

S7CPU 提供延时 OB 用于在用户程序中编写延时执行的程序。使用延时中断可以获得精度较高的延时，延时中断以毫秒（ms）为单位定时。各 CPU 可以使用的延时中断 OB（OB20 ~ OB23）的个数与 CPU 的型号有关，S7-300CPU（不包括 CPU318）只能使用 OB20。延时中断 OB 优先级的默认设置值为 3 ~ 6 级。延时中断 OB 用 SFC32“SRT_ DINT”起动，延时时间在 SFC32 中设置，起动后经过设定的延时时间，触发中断，调用 SFC32 指定的 OB。需要延时执行的操作放在 OB 中，必须将延时中断 OB 作为用户程序的一部分下载到 CPU。

如果延时中断已被起动，延时时间还没有到达，可以用 SFC33“ CAN_DINT”取消延时中断的执行。SFC34“QRY_DINT”用来查询延时中断的状态。表 6-9 给出了 SFC34 输出的状态字节 STATUS。

只有在 CPU 处于运行状态时才能执行延时中断 OB，暖起动或冷起动都会清除延时中断 OB 的起动事件。

表 6-9　SFC 34 输出的状态字节 STATUS

位	取值	意　义	位	取值	意　义
0	0	延时中断已被允许	3	0	—
1	0	未拒绝新的延时中断	4	0	没有装载延时中断 OB
2	0	延时中断未被激活或时间已过去	5	0	日期时间中断 OB 的执行没有被激活的测试功能禁止

五、循环中断组织块

S7CPU 提供循环中断 OB，可用于按一定时间间隔中断循环程序的执行，例如周期性地定时执行闭环控制系统的 PID 运算程序，间隔时间从 STOP 切换到 RUN 模式时开始计算。

用户定义在时间间隔时，必须确保在两次循环中断之间的时间间隔中且有足够的时间处理循环中断程序。

各 CPU 可以使用的循环中断 OB（OB30 ~ OB38）的个数与 CPU 的型号有关，S7-300 CPU（不包括 CPU 318）只能使用 OB35。OB30 ~ OB38 默认的时间间隔和中断优先级见表 6-10 。如果两个 OB 的时间间隔成整倍数，不同的循环中断 OB 可能同时请求中断，造成处理循环中断服务程序的时间超过指定的循环时间。为了避免出现这样的错误，用户可以定义一个相位偏移。相位偏移用于在循环时间间隔到达时，延时一定的时间后再执行循环中断。相位偏移 m 的单位为 ms，应有 $0 < m < n$ ，式中 n 为循环的时间间隔。

没有专用的 SFC 来激活和禁止循环中断，可以用 SFC40 和 SFC39 来激活和禁止它们。SFC40 “EN_INT” 是用于激活新的中断和异步错误的系统功能，其参数 MODE 为 0 时激活所有的中断和异步错误，为 1 时激活部分中断和错误，为 2 时激活指定的 OB 编号对应的中断和异步错误。SFC39 “DIS_INT” 是禁止新的中断和异步错误的系统功能，MODE 为 2 时禁止指定的 OB 编号对应的中断和异步错误，MODE 必须用十六进制数来设置。

表 6-10　循环 OB 默认参数

OB 号	时间间隔	优先级	OB 号	时间间隔	优先级
OB30	5s	7	OB35	100ms	12
OB31	2s	8	OB36	50ms	13
OB32	1s	9	OB37	20ms	14
OB33	500ms	10	OB38	10ms	15
OB34	200ms	11			

六、硬件中断组织块

S7 CPU 提供硬件中断 OB（OB40 ~ OB47），用于对模板（如信号模板（SM）、通信处理器（CP）和功能模块（FM））上的信号变化进行快速响应。

硬件中断被模块触发后，操作系统将自动识别是哪一个槽的模块和模块中哪一个通道产生的硬件中断。硬件中断 OB 执行完后，将发送通道确认信号。硬件中断信号的处理如图 6-15 所示。

硬件中断 OB 的默认优先级为 16 ~ 23，用户可以设置参数改变优先级。

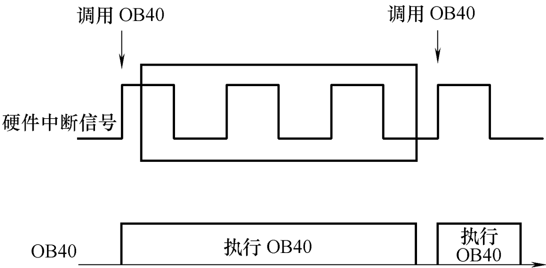


图 6-15　硬件中断信号的处理

如果在处理硬件中断的同时，又出现了其他硬件中断事件，新的中断按以下方法识别和处理：如果正在处理某一中断事件，又出现了同一模块同一通道产生的完全相同的中断事件，新的中断事件将丢失，即不处理它。在图 6-15 数字量模块输入信号的第一个上升沿时触发中断，由于正在用 OB40 处理中断，第二和第三上升沿产生的中断信号丢失。

如果正在处理某一中断事件，又出现了同一模块同一通道产生的完全相同的中断事件，新的中断事件将丢失。

如果正在处理某一中断信号时同一模块中其他通道产生了中断事件，新的中断不会被立即触发，但是不会丢失。在当前已激活的硬件中断执行完后，再处理被暂存的中断。如果硬件中断被触发，并且它的 OB 其他模块中的硬件中断激活，新的请求将被记录，空闲后再执行该中断。

七、背景组织块

CPU 可以保证设置的最小扫描循环时间，如果它比实际的扫描循环时间长，在循环程序结

束后 CPU 处于空闲的时间内可以执行背景 OB（OB90）。如果没有对 OB90 编程，CPU 等到定义的最小扫描循环时间到达为止，再开始下一次循环的操作。用户可以将对运行时间要求不高的操作放在 OB90 中去执行，以避免出现等待时间。背景 OB 优先级为 29（最低），不能通过参数设置进行修改。OB90 可以被所有其他的系统功能和任务中断。

由于 OB90 运行时间不受 CPU 操作系统的监视，用户可以在 OB90 中编写长度不受限制的程序。

八、起动组织块 OB100/OB101/OB102

1. CPU 模块的起动类型

（1）暖起动（Warm Restart）

暖起动时，过程映像数据以及非保持的存储器位、定时器和计数器被复位。具有保持功能的存储器位、定时器、计数器和所有数据块将保留原数值。程序将重新开始运行，执行起动 OB 或 OB1。S7-300 CPU（不包括 CPU 318）只有暖起动。

手动暖起动时，将模式选择开关扳到 STOP 位置，“STOP” LED 亮，然后扳到 RUN 或 RUN-P 位置。

（2）热起动（Hot Restart）

在 RUN 状态时如果电源突然丢失，然后又重新上电，S7-400 CPU 将执行一个初始化程序，自动地完成热起动。热起动从上次 RUN 模式结束时程序被中断之处继续执行，不对计数器等复位。热起动只能在 STOP 状态时没有修改用户程序的条件下才能进行。热起动仅在 S7-400 中有。

（3）冷起动（Cold Restart）

冷起动时，过程数据区的所有过程映像数据、存储器位、定时器、计数器和数据块均被清除，即被复位为零，包括有保持功能的数据。用户程序将重新开始运行，执行起动 OB 和 OB1。

手动冷起动时将模式选择开关扳到 STOP 位置，“STOP” LED 亮，再扳到 MRES 位置，“STOP” LED 灭 1s，亮 1s，再灭 1s 后保持亮。最后将它扳到 RUN 或 RUN-P 位置。

2. 起动组织块（OB100 ~ OB102）

下列事件发生时，CPU 执行起动功能：

1）PLC 电源上电后。

2）CPU 的模式选择开关从 STOP 位置扳到 RUN 或 RUN-P 位置。

3）接收到通过通信功能发送来的起动请求。

4）多 CPU 方式同步之后和 H 系统连接好后（只适用于备用 CPU）。

起动用户程序之前，应先执行起动 OB。在暖起动、热起动或冷起动时，操作系统分别调用 OB100、OB101 或 OB102，S7-300 和 S7-400H 不能热起动。

用户可以通过在起动组织块 OB100 ~ OB 102 中编写程序，来设置 CPU 的初始化操作，例如开始运行的初始值，I/O 模块的初始值等。

起动程序没有长度和时间的限制，因为循环时间监视还没有被激活，在起动程序中不能执行时间中断程序和硬件中断程序。

CPU 318-2 只允许手动暖起动或冷起动。对于某些 S7-400 CPU，如果允许用户通过 STEP7 的参数设置手动起动，用户可以使用状态选择开关和起动类型开关（CRST/WRST）进行手动起动。

在设置 CPU 模块属性的对话框中，选择“Startup”选项卡，可以设置起动的各种参数。起动 S7-400 CPU 时，作为默认的设置，将输出过程映像区清零。如果用户希望在起动之后继续在用户程序中使用原有的值，也可以选择不将过程映像区清零。

为了在起动时监视是否有错误，用户可以选择以下的监视时间：

1）向模块传递参数的最大允许时间。

2）上电后模块向 CPU 发送“准备好”信号允许的最大时间。

3）S7-400 CPU 热起动允许的最大时间，即电源中断的时间或由 STOP 转换为 RUN 的时间。一旦超过监视时间，CPU 将进入停机状态或只能暖起动。如果监控时间设置为 0，表示不监控。

九、故障处理组织块

1. 错误处理概述

S7-300/400 有很强的故障检测和处理能力。这里所说的错误指的是 PLC 内部的功能性错误或编程错误，而不是外部设备的故障。CPU 检测到错误后，操作系统调用对应的组织块，用户可以在组织块中编程，对发生的错误采取相应的措施。对于大多数错误，如果没有给组织块编程，出现错误时 CPU 将进入 STOP 模式。

系统程序可以检测出下列错误：不正确的 CPU 功能、系统程序执行中的错误、用户程序中的错误和 I/O 中的错误。根据错误类型的不同，CPU 被设置为进入 STOP 模式或调用一个错误处理 OB。

当 CPU 检测到错误时，会调用适当的故障处理组织块（见表 6-11），如果没有相应的错误处理 OB，CPU 将进入 STOP 模式。用户可以在错误处理 OB 中编写如何处理这种错误的程序，以减小或消除错误的影响。为避免发生某种错误时 CPU 进入停机状态，可以在 CPU 中建立一个对应的空的组织块。

表 6-11 错误处理组织块

OB 号	错 误 类 型	优先级	OB 号	错 误 类 型	优先级
OB70	I/O 冗余错误（仅 H 系列 CPU）	25	OB84	CPU 硬件故障	26/28
OB72	CPU 冗余错误（仅 H 系列 CPU）	28	OB85	优先级错误	
OB73	通信冗余错误（仅 H 系列 CPU）	25	OB86	机架故障或分布式 I/O 的站故障	
OB80	时间错误	26	OB87	通信错误	
OB81	电源故障	26/28	OB121	编程错误	引起错误的 OB 的优先级
OB82	诊断中断		OB122	I/O 访问错误	
OB83	插入/取出模块中断				

2. 错误的分类

被 S7-CPU 检测到，并且用户可以通过组织块对其进行处理的错误，可分为两个基本类型。

（1）异步错误

异步错误是与 PLC 的硬件或操作系统密切相关的错误，它们不会出现在用户程序的执行过程中。该类错误可能是优先级错误、PLC 故障或冗余错误，后果较严重。异步错误 OB 具有最高等级的优先级，其他 OB 不能中断它们。同时有多个相同优先级的异步错误 OB 出现，将按出现的顺序处理。

（2）同步错误（OB121 和 OB122）

同步错误是与程序执行有关的错误，其 OB 的优先级与出现错误时被中断的块的优先级相同，即同步错误 OB 中的程序可以访问块被中断时累加器和状态寄存器中的内容。对错误进行处理后，可以将处理结果返回被中断的块。

3. 电源故障处理组织块（OB81）

电源故障包括后备电池失效或未安装，S7-400 的 CPU 机架或扩展机架上的 DC 24V 电源故障。电源故障出现和消失时操作系统都要调用 OB81。

4. 时间错误处理组织块（OB80）

循环监控时间的默认值为 150ms，时间错误包括实际循环时间超过设置的循环时间、因为向

前修改时间而跳过日期时间中断、处理优先级时延迟太多等。

5. 诊断中断处理组织块（OB82）

如果模块有诊断功能并且激活了它的诊断中断，当它检测到错误时，以及错误消失时，操作系统都会调用 OB82。

OB82 在下列情况时被调用：有诊断功能的模块的断线故障，模拟量输入模块的电源故障，输入信号超过模拟量模块的测量范围等。用 SFC 51“RDSYSST”可以读出模块的诊断数据。用 SFC52“WR_USMSG”可以将这些信息存入诊断缓冲区，也可以发送一个用户定义的诊断报文到监控设备。

6. 插入/拔出模块中断组织块（OB83）

S7-400 可以在 RUN、STOP 或 STARTUP 模式下带电拔出和插入模块，但是不包括 CPU 模块、电源模块、接口模块和带适配器的 S5 模块，上述操作将会产生插入/拔出模块中断。

7. CPU 硬件故障处理组织块（OB84）

当 CPU 检测到 MPI 网络的接口故障、通信总线的接口故障或分布式 I/O 网卡的接口故障时，操作系统调用 OB84。故障消除时也会调用该 OB。

8. 优先级错误处理组织块（OB85）

在以下情况下将会触发优先级错误中断：

1）产生了一个中断事件，但是对应的 OB 没有下载到 CPU。

2）访问一个系统功能块的背景数据块时出错。

3）刷新过程映像表时 I/O 访问出错，模块不存在或有故障。

9. 机架故障组织块（OB86）

在以下情况下将会触发机架故障中断：

1）机架故障，例如找不到接口模块或接口模块损坏，或者连接电缆断线。

2）机架上的分布式电源故障。

3）在 SINEC L2-DP 总线系统的主系统中有一个 DP 从站有故障。

10. 通信错误组织块（OB87）

在以下情况下将会触发通信错误中断：

1）接收全局数据时，检测到不正确的帧标识符（ID）。

2）全局数据通信的状态信息数据块不存在或太短。

3）接收到非法的全局数据包编号。

十、同步错误组织块

1. 同步错误

同步错误发生在执行某一特定指令的过程中，是与执行用户程序有关的错误，程序中如果有不正确的地址区、错误的编号或错误的地址，都会出现同步错误，操作系统将调用同步错误 OB。OB121 用于对程序错误的处理，OB122 用于处理模块访问错误。

同步错误 OB 的优先级与检测到出错的块的优先级一致。因此 OB121 和 OB122 可以访问中断发生时累加器和其他寄存器中的内容。用户程序可以用它们来处理错误，例如出现对某个模拟量输入模块的访问错误时，可以在 OB122 中用 SFC 44 定义一个替代值。同步错误可以用 SFC 36“MASK_FLT”来屏蔽，使某些同步错误不触发同步错误 OB 的调用，但是 CPU 会在错误寄存器中记录发生的被屏蔽的错误，并用错误过滤器中的一位来表示某种同步错误是否被屏蔽。错误过滤器分为程序错误过滤器和访问错误过滤器，分别占一个双字。调用 SFC 37“DMSK_FLT”并且在当前优先级被执行完后，将解除被屏蔽的错误，可以用 SFC 38“READ_ERR”读出已经发生

的被屏蔽的错误。

对于 S7-300（ CPU318 除外），不管错误是否被屏蔽，错误都会被送入诊断缓冲区，并且 CPU 的“组错误”LED 会被点亮。

2. 编程错误组织块（OB121）

出现编程错误时，CPU 的操作系统将调用 OB121。局域变量 OB121_SW_FLT 给出了错误代码，可以查看《S7-300/400 的系统软件和标准功能》中 OB121 部分的错误代码表。

3. I/O 访问错误组织块（OB122）

STEP 7 指令访问有故障的模块，例如直接访问 I/O 错误（模块损坏或找不到），或者访问了一个 CPU 不能识别的 I/O 地址，此时 CPU 的操作系统将会调用 OB122。

十一、常用 OB 组织块的使用举例

现以 CPU315（6ES7　315-2AG10-0AB0），STEP7 V5.3 为例介绍常用 OB 的使用方法，这些 OB 包括：

程序循环组织块（OB1）；

日期时间中断组织块（以 OB10 为例）；

延时中断组织块（以 OB20 为例）；

循环中断组织块（以 OB35 为例）；

硬件中断组织块（以 OB40 为例）；

诊断中断组织块（以 OB82 为例）；

机架故障组织块（以 OB86 为例）；

起动的类型（CPU300 以 OB100 为例，CPU400 以 OB101、OB102 为例）；

编程故障组织块（以 OB121 为例）；

I/O　访问故障组织块（以 OB122 为例）。

还有其他的 OB，如：I/O 冗余故障 OB（OB70），CPU 冗余故障 OB（OB72），通信冗余故障 OB（OB73），请咨询 CPU400H 系统工程师，这里不做说明。

1. 程序循环组织块（OB1）

（1）硬件组态

在“OB_Example”项目中插入一 S7300 站，命名为“OB1_Example”，然后插入“CPU 315-2DP”，如图 6-16 所示。

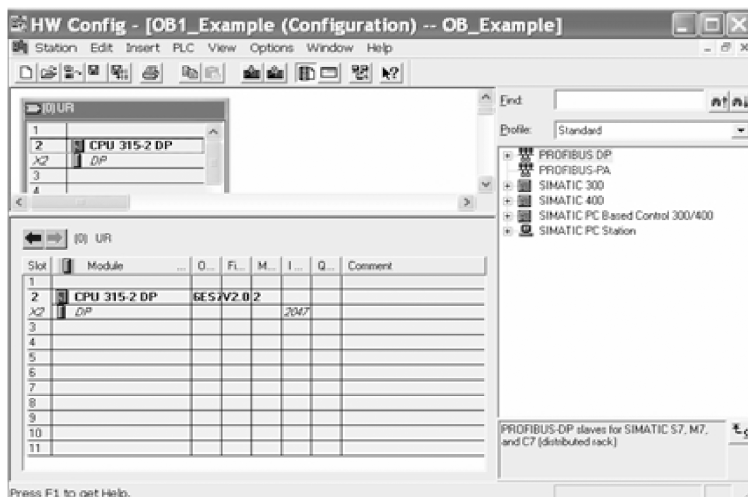


图 6-16　硬件组态

硬件组态完成后，保存编译。

（2）OB1 程序执行

OB1 的程序循环执行，用 STEP7 可以时时监控程序的运行，具体程序参见 OB_Example/OB1_Example。OB1 的 STL 程序（可转成梯形图）为：

NetWork1：

L MB 100

T MB 0

NOP 0

将 OB1 程序和硬件组态下载到 CPU 中。其中，MB100 为时钟存储器，设置方法为进入硬件组态，双击“CPU315-2DP”，选择“Cycle/Clock Memory”，具体设置画面如图 6-17 所示。

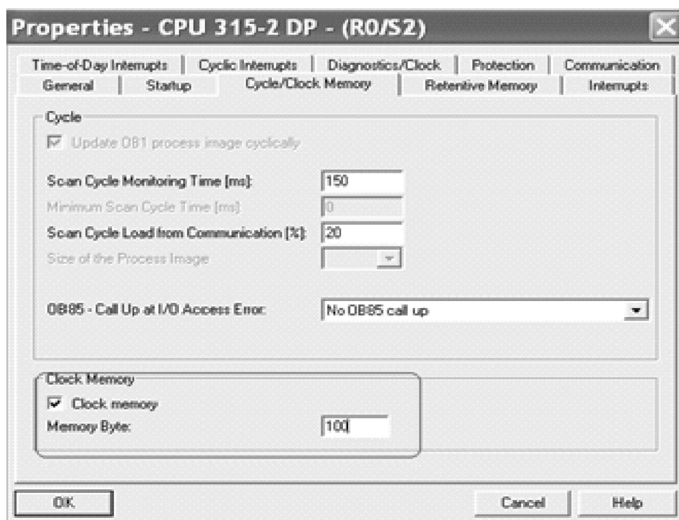


图 6-17 设置画面

STEP7 时时监控画面如图 6-18 所示。

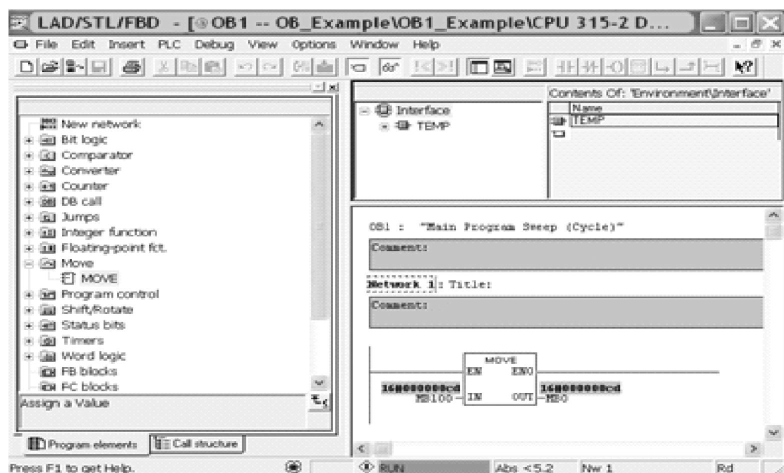


图 6-18 STEP7 时时监控画面

2. 日期时间中断组织块（OB10）

（1）硬件组态在“OB_Example”项目中插入一 S7300 站，命名为“OB10_Example”，然后插入“CPU 315-2DP”，如图 6-19 所示。

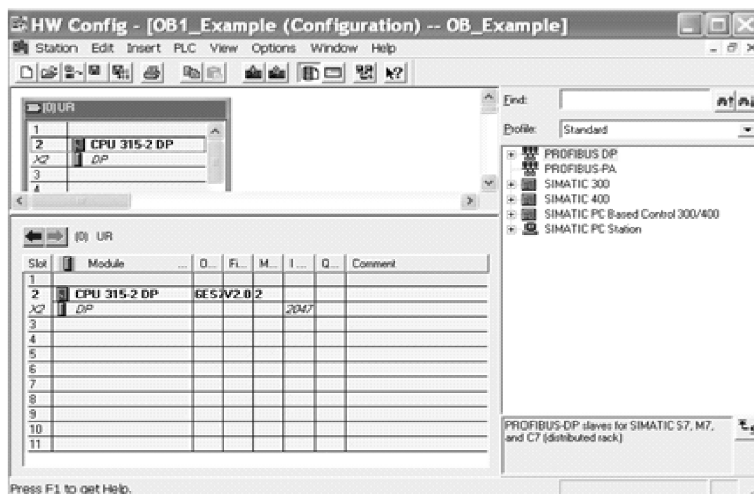


图 6-19 硬件组态

双击“CPU 315-2DP”，选择“Time-of-Day Interrupts”选项，选中“Active”，同时设置“Execution”选项，本例选择“Every minute”，“Execution”选项包括：

None：不使用；

Once：只执行一次；

Every minute：每分钟执行一次；

Every hour：每小时执行一次；

Every week：每周执行一次；

Every month：每月执行一次；

End of month：月末执行一次；

Every year：每年执行一次。

设置开始执行的日期（Start date）和时间（Time of day），设置完成后的画面如图 6-20 所示。

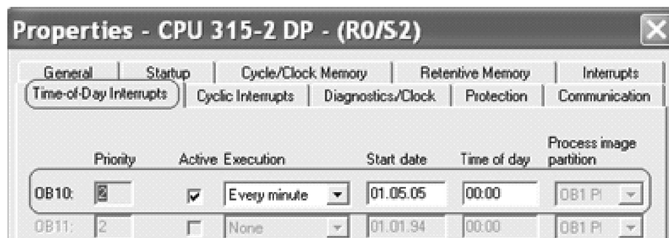


图 6-20 设置完成后的画面

硬件组态完成后，保存编译。

（2）OB10 程序执行

OB10 程序按照设定的时间执行，使用 STEP7 不能时时监控程序的运行，可用 Variable Table 监控实时数据变化。具体程序参见 OB_Example/OB10_Example。

在 OB10_Example 程序的 Blocks 中插入组织块 OB10，如图 6-21 所示。

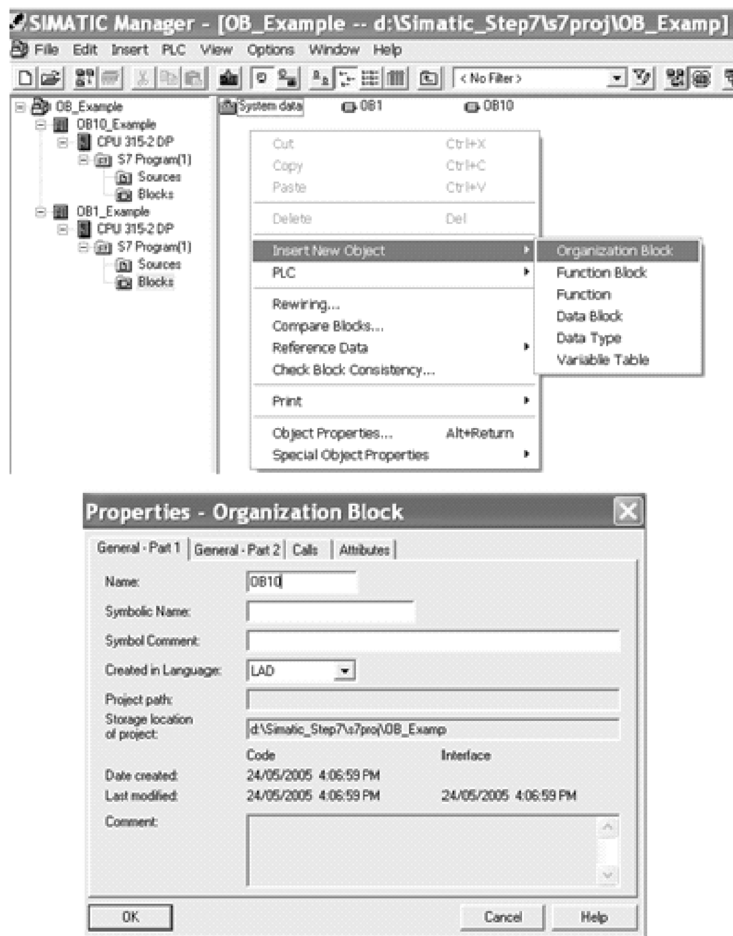


图 6-21　插入组织块 OB10

然后打开组织块 OB10 编写程序，OB10 的 STL 程序（可转成梯形图）为：

```
NetWork1：
L   MW   0
L   1
+I
T   MW   0
NOP   0
```

将 OB10 程序和硬件组态下载到 CPU 中。

在 OB10_Example 程序的 Blocks 中插入 Variable Table，然后打开，填入地址 MW0 并点击“Monitor Variable”按钮，设置画面如图6-22所示。

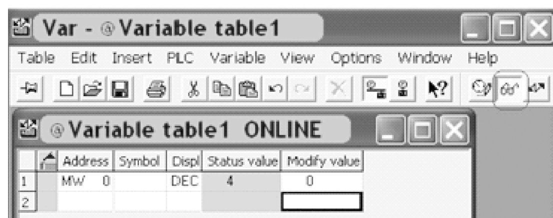


图 6-22　设置画面

此时可以监控 MW0 每分钟加 1。

3. 延时中断组织块（OB20）

（1）硬件组态

在“OB_Example”项目中插入一 S7300 站，命名为“OB20_Example”，然后插入

“CPU 315-2DP”，如图 6-23 所示。

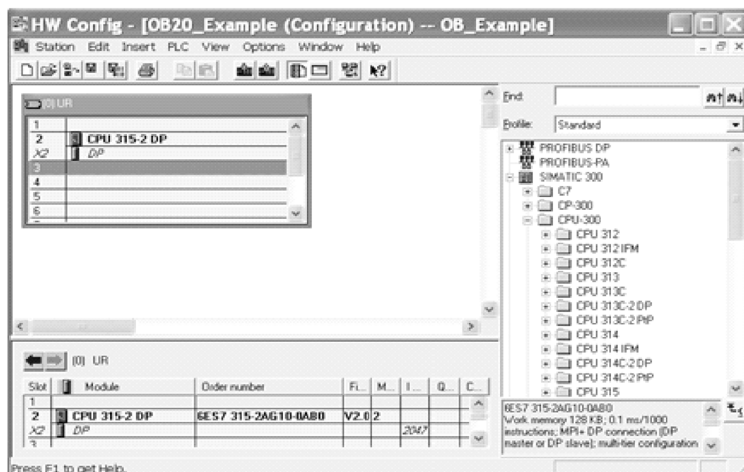


图 6-23　硬件组态

双击“CPU 315-2DP”，选择“Interrupts”选项，可以看到 CPU 支持 OB20，组态完的画面如图 6-24 所示。

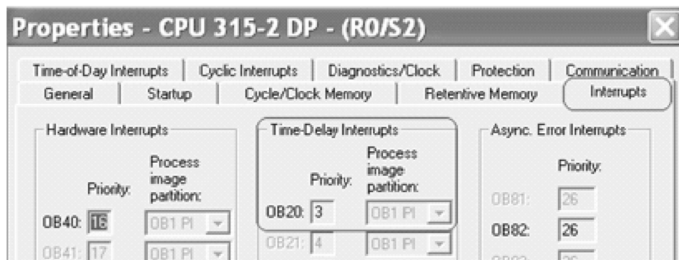


图 6-24　组态完的画面

硬件组态完成后，保存编译。

（2）OB20 程序执行

每一次 OB20 的程序执行，必须调用 SFC32（SRT_DINT），延迟时间在 SFC 的输入参数中给定，同时给定 OB 号，调用 SFC32 且设定的时间延迟到后，执行 OB 程序，如果再次执行 OB 程序，需要再次调用 SFC32。如果在延迟时间未到之前想取消程序的执行，可以调用 SFC33（CAN_DINT），同时可以使用 SFC34（QRY_DINT）取得延迟中断的状态，具体的 SFC32/33/34 的调用方法可 参考在线帮助，STEP7 不能时时监控程序的运行，可用 Variable Table 监控实时数据变

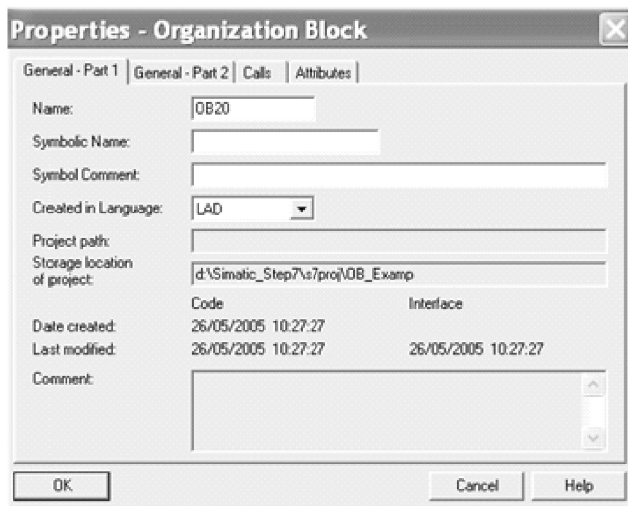


图 6-25　插入组织块 OB20

化。具体程序参见 OB_Example/OB20_Example。在 OB20_Example 程序的 Blocks 中插入组织块 OB20，如图 6-25 所示。

然后打开组织块 OB20 编写程序，OB20 的 STL 程序（可转成梯形图）为：

```
NetWork1：
L   MW   0
L   1
 +I
T   MW   0
NOP   0
```

打开组织块 OB1 编写程序，OB1 的 STL 程序（可转成梯形图）为：

```
NetWork1：
A   M   20.0
JNB   _001
CALL   " SRT_DINT"
OB_NR   ：  =20
DTIME   ：  =T#10S
SIGN   ：  =MW10
RET_VAL：  =MW12
_001：   A   BR
R   M   20.0
NetWork2：
A   M   20.1
JNB   _002
CALL      " CAN_DINT"
OB_NR：  =20
RET_VAL：  =MW14
_002：   A   BR
R   M   20.1
NetWork3：
CALL      " QRY_DINT"
OB_NR   ：  =20
RET_VAL：  =MW16
STATUS   ：  =MW18
NOP   0
```

将 OB1、OB20 和硬件组态下载到 CPU 中。在 OB20_Example 程序的 Blocks 中插入 Variable Table，然后打开，填入地址 MW0、M20.0、M20.1、MW18 并点击“Monitor Variable”按钮，设置画面如图 6-26 所示。

此时可以监控 MW0 的变化，将 M20.0 置为 true，10s 后延迟时间到，MW0 加 1，再将 M20.0 置为 true，10s 后延迟时间到，MW0 再加 1。如果当延迟时间未到，此时将 M20.1 置为 true，那么此次时间延迟中断被取消，MW0 不会加 1，每次执行的状态都可以从 MW18 中读出，具体状态的含义请参阅 SFC34（QRY_DINT）的在线帮助。

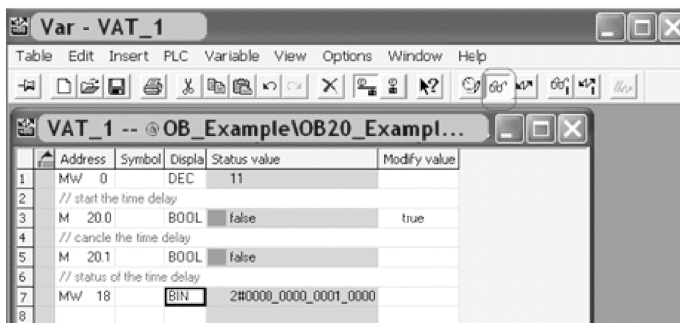


图 6-26　设置画面

4. 循环中断组织块（OB35）

（1）硬件组态

在“OB_Example”项目中插入一 S7300 站，命名为“OB35_Example”，然后插入“CPU 315-2DP”，参见 OB10 硬件组态，双击“CPU 315-2DP”，选择“Cyclic Interrupts”选项，修改 OB35 的执行周期（Execution（ms），范围是 1～60000ms），本例设为 1000ms，如图 6-27 所示。

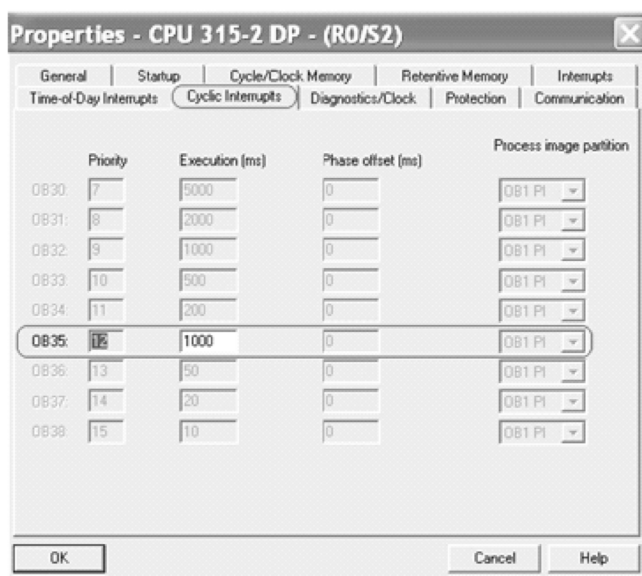


图 6-27　硬件组态

硬件组态完成后，保存编译。

（2）OB35 程序执行

OB35 程序按照设定的执行周期循环执行，使用 STEP7 不能时时监控程序的运行，可用 Variable Table 监控实时数据变化。具体程序参见 OB_Example/OB35_Example。

在 OB35_Example 程序的 Blocks 中插入组织块 OB35，如图 6-28 所示。

然后打开组织块 OB35 编写程序，OB35 的 STL 程序（可转成梯形图）为：

```
NetWork1：
L   MW   0
L   1
 +I
T   MW   0
NOP   0
```

将 OB351 和硬件组态下载到 CPU 中。在 OB35_Example 程序的 Blocks 中插入 Variable Table，然后打开，填入地址 MW0 并点击“Monitor Variable”按钮，设置画面如图 6-29 所示。

此时可以监控 MW0 每秒钟加 1。可以在 OB35 中周期性的调用 PID 模块（FB41/42/43），完成 PID 调节，也可以 OB35 中调用周期的数据发送指令，完成数据发送功能，等等，总之，OB35

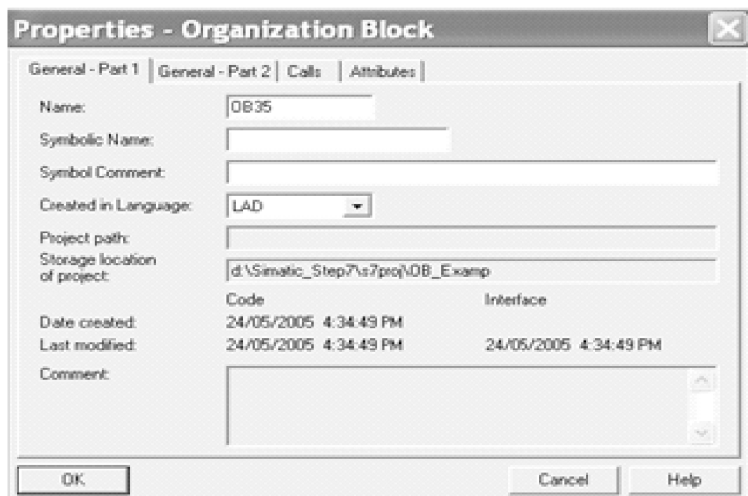


图 6-28　插入组织块 OB35

是按设定的循环周期执行。

5. 硬件中断组织块（OB40）

（1）硬件组态

在“OB_Example”项目中插入一 S7300 站，命名为“OB40_Example”，然后插入“CPU 315-2DP”和一块具有中断功能的数字量输入模板“6ES7 321-7BH01-0AB0 ”。

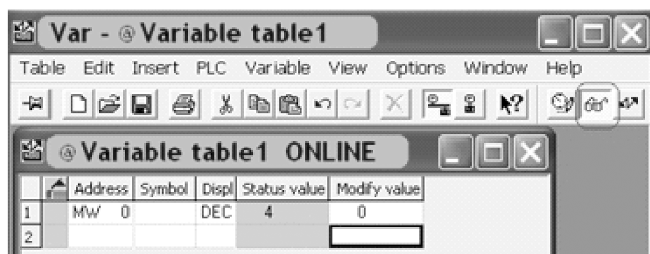


图 6-29　设置画面

双击“6ES7 321-7BH01-0AB0”模板，选择“Inputs”选项，同时选中“Hardware interrupt”和“Trigger for Hardware Interrupt”选项，如图 6-30 所示。

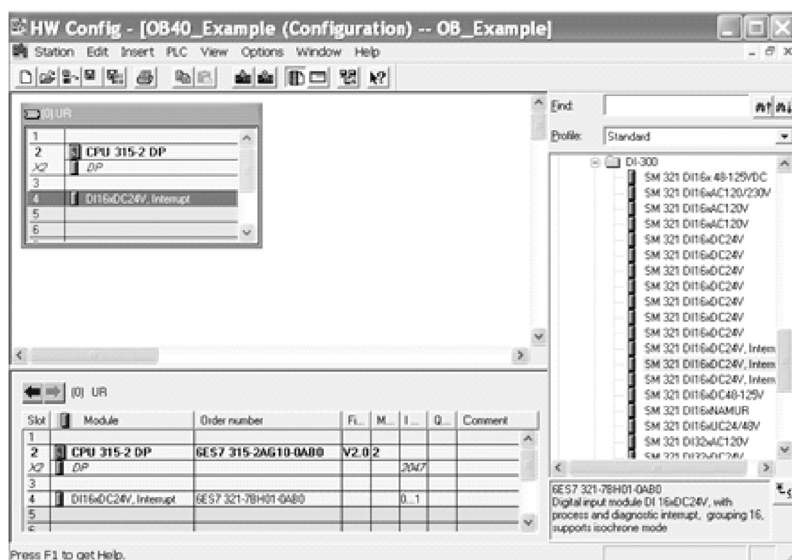


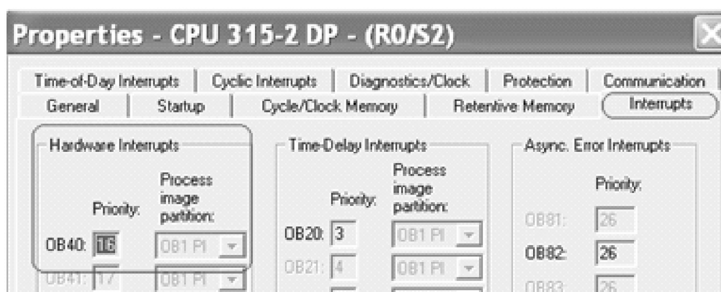
图 6-30　硬件组态

点击“OK”，然后双击“CPU315-2DP”，选择“Interrupts”选项，可以看到 CPU 支持 OB40，如图 6-31 所示。

硬件组态完成后，保存编译。

（2）OB40 程序执行

OB40 程序在硬件组态中设定的硬件中断发生后执行，当 OB40 执行时可以通过它的临时变量 OB40_MDL_ADDR 读出产生硬件中断的模板的逻辑地址，通过 OB40_POINT_ADDR 可以读出产生硬件中断的通道，临时变量的具体含义请参阅在线帮助。STEP7 不能时时监控程序的运行，可用 Variable Table 监控实时数据变化。具体程序参见 OB_Example/OB40_Example。



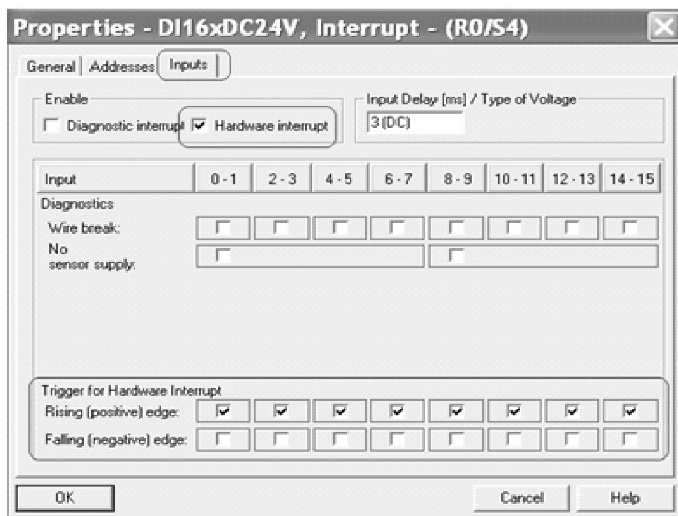


图 6-31　组态完成画面

在 OB40_Example 程序的 Blocks 中插入组织块 OB40，如图 6-32 所示。

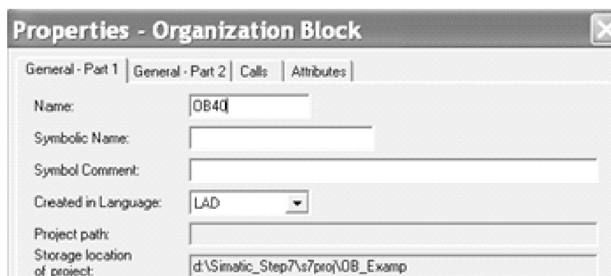


图 6-32　插入组织块 OB40

然后打开组织块 OB40 编写程序，OB40 的 STL 程序（可转成梯形图）为：

```
NetWork1:
L   MW   0
L   1
 +I
T   MW   0
NOP   0
NetWork2:
A(
L   #OB40_MDL_ADDR
T   MW   10
SET
SAVE
CLR
A   BR
)
JNB_001
```

L #OB40_POINT_ADDR

T MD 12

_001： NOP 0

将 OB40 和硬件组态下载到 CPU 中。

在 OB40_Example 程序的 Blocks 中插入 Variable Table，然后打开，填入地址 MW0、MW10、MD12 并点击“Monitor Variable”按钮，设置画面如图 6-33 所示。

此时可以监控 MW0 的变化，每当 I0.1 有上升沿脉冲产生 MW0 加 1，MW10 为硬件中断模板的逻辑地址，本例中为 0，MD12 为中断产生的通道号，注意此值以十六进制表示。

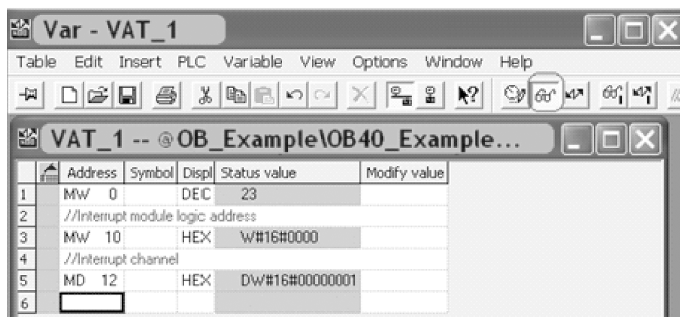


图 6-33 设置画面

6. 诊断中断组织块（OB82）

结合模板的断线检测应用和 SFC51 来说明诊断中断组织块 OB82 的使用方法。

（1）硬件组态

在“OB_Example”项目中插入一 S7300 站，命名为“OB82_Example”，然后插入“CPU 315-2DP”和一块具有中断功能的模拟量输入模板“6ES7 331-7KF02-0AB0”，配置“SM331-7KF02-0AB0”模块的“Inputs”选项，选择 0-1 通道组为 2 线制电流（2DMU），其他通道组为电压，并注意模板的量程卡与设置的相同。选中“Enable”框中的“Diagnostic Interrupt”选项，选中“Diagnostics”选项中的 0-1 通道组中的“Group Diagnostics”和“with Check for Wire Break”选项，配置完成的画面如图 6-34 所示。

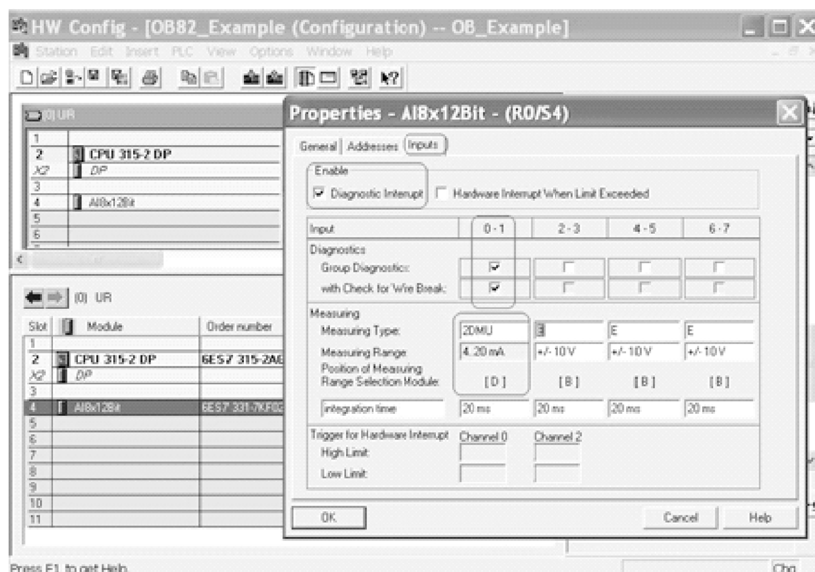


图 6-34 配置完成的画面

点击“OK”，然后双击“CPU315-2DP”，选择 Interrupts 选项，可以看到 CPU 支持 OB82，如图 6-35 所示。

硬件组态完成后，保存编译。

（2）OB82 程序执行

OB82 程序在硬件组态中设定的诊断中断发生后执行，当 OB82 执行时可以通过它的临时变量 OB82_MDL_ADDR 读出产生诊断中断的模板的逻辑地址，OB82 其他临时变量的具体含义请参阅 OB82 的在线帮助。STEP7 不能时时监控程序的运行。接下来完成诊断程序。

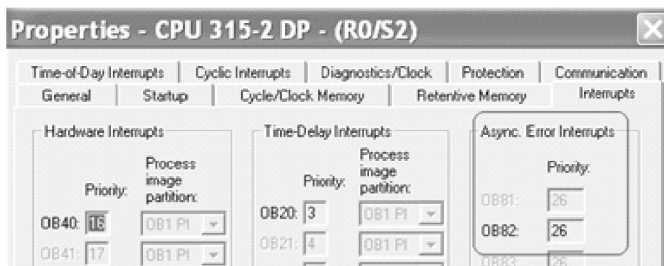


图 6-35　组态完成画面

1）在 OB_Example/OB82_Example/CPU315-2DP/S7 Program（7）/Sources 下面插入 STL Source 文件 STL Source（1）。

2）打开空的 OB1 程序，然后选中 Libraries→Standard Libraries→System Function Blocks→SFC51 RDSYSST DIAGNSTC，按“F1”键，出现 SFC51 的在线帮助信息。可具体读一下信息的内容，然后在信息的最底部点击“Example for module diagnostics with the SFC 51”，然后点击“STL Source File”，选中全部 STL Source 源程序复制到 STL Source（1）中，存盘编译此源程序，提示没有错误。

3）在 Blocks 中生成 OB1、OB82、DB13 和 SFC51。

4）打开 OB82 的程序并做简单修改，将 19 和 20 行复制到 go：后面并保存，具体变化如图 6-36 所示。

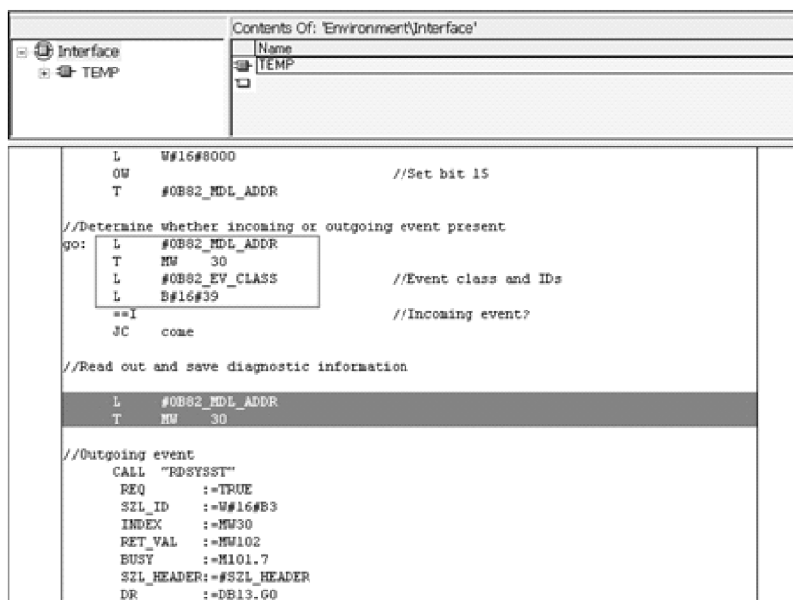


图 6-36　具体变化

5）将整个 S7-300 站的程序和硬件组态下载到 CPU 中。下载完成后，将 CPU 的模式选择开关切换到 RUN 位置，此时 CPU“RUN”灯亮、“SF”灯亮，SM331 的“SF”灯亮。同时，查看 CPU 的诊断缓冲区 Hardware→Online，双击 CPU，选择“Diagnostic Buffer”，可获得相应的故障信息。

6）打开数据块 DB13，在线监控，具体画面如图 6-37 所示。

Address	Name	Type	Initial value	Actual value	Comment
0.0	COME[1]	BYTE	B#16#0	B#16#0D	
1.0	COME[2]	BYTE	B#16#0	B#16#15	
2.0	COME[3]	BYTE	B#16#0	B#16#00	
3.0	COME[4]	BYTE	B#16#0	B#16#00	
4.0	COME[5]	BYTE	B#16#0	B#16#71	
5.0	COME[6]	BYTE	B#16#0	B#16#08	
6.0	COME[7]	BYTE	B#16#0	B#16#08	
7.0	COME[8]	BYTE	B#16#0	B#16#03	
8.0	COME[9]	BYTE	B#16#0	B#16#10	
9.0	COME[10]	BYTE	B#16#0	B#16#10	
10.0	COME[11]	BYTE	B#16#0	B#16#00	
11.0	COME[12]	BYTE	B#16#0	B#16#00	
12.0	COME[13]	BYTE	B#16#0	B#16#00	
13.0	COME[14]	BYTE	B#16#0	B#16#00	
14.0	COME[15]	BYTE	B#16#0	B#16#00	
15.0	COME[16]	BYTE	B#16#0	B#16#00	

图 6-37　数据块 DB13 的在线监控画面

因为通道断线是一个到来事件，所以诊断信息存储到 COME 数组中，具体每一字节的含义参见 S7-300 的硬件手册中 B Diagnostics Data of Signal Modules 部分的详细说明，S7-300 的硬件手册可以从西门子网站下载，下载网址为：http：//support. automation. siemens. com/WW/view/en/8859629。

7）本例中 COME 数组字节的含义解释如下：

COME ［1］=16#0D　表示通道错误，外部故障和模板问题；

COME ［2］=16#15　表示此段信息为模拟量模板的通道信息；

COME ［3］=16#00　表示 CPU 处于运行状态，无字节 2 中标示的故障信息；

COME ［4］=16#00　表示无字节 3 中标示的故障信息；COME ［5］=16#71 表示模拟量输入；

COME ［6］=16#08　表示模板的每个通道有 8 个诊断位；COME ［7］=16#08 表示模板的通道数；

COME ［8］=16#03　表示 0 通道错误和 1 通道错误，其他通道正常；

COME ［9］=16#10　表示 0 通道断线；

COME ［10］=16#10　表示 1 通道断线；

COME ［11］=16#00　表示 2 通道正常，其他通道与 2 通道相同。

8）如何读取其他信息的诊断可详细参考 OB82、SFC51 和 S7-300 的硬件手册中 B / Diagnostics Data of Signal Modules 部分的说明。

7. 机架故障组织块（OB86）

（1）硬件组态

在“OB_Example”项目中插入一 S7300 站，命名为“OB86_Example”，然后插入“CPU 315-2DP”，选择 DP 作为主站，在 DP 主站下面添加一 ET200M 从站，并在从站中插入一模拟量模块 SM331（6ES7331-7KF02-0AB0），同时注意 CPU 的 DP 主站地址和 ET200M 从站地址不能相同，并且 ET200M 的站地址必须和 ET200M 上的实际地址一致，组态完成后的画面如图 6-38 所示。

然后双击“CPU315-2DP”，选择“Interrupts”选项，可以看到 CPU 支持 OB86，画面如图 6-39所示。

硬件组态完成后，保存编译。

（2）OB86 程序执行

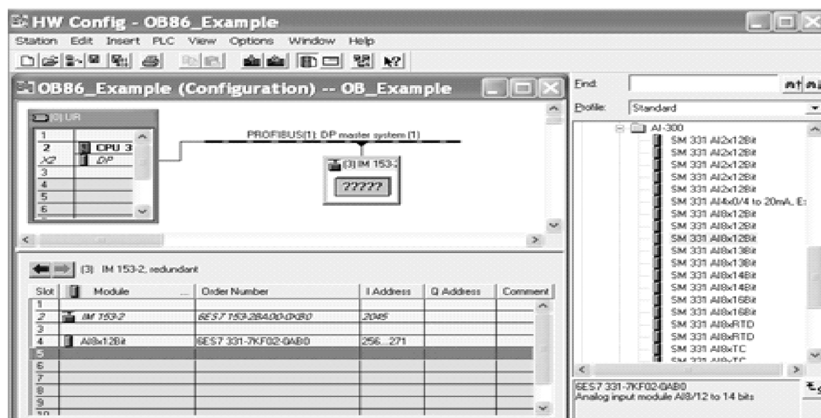


图 6-38　组态完成后的画面

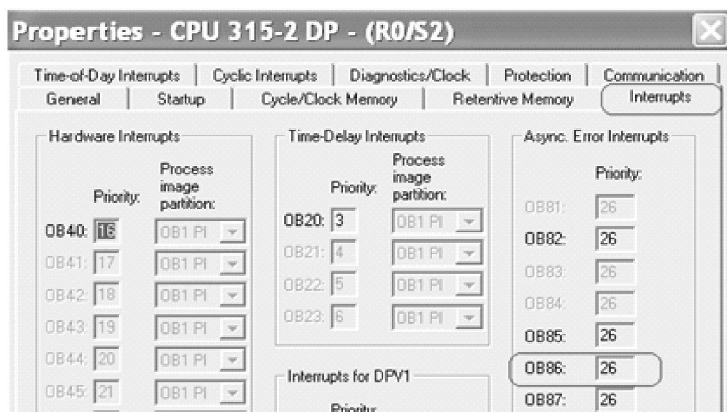


图 6-39　组态完成画面

OB86 程序在通信发生问题后或者访问不到配置的机架或站时执行，此时程序还可能需要调用 OB82 和 OB122 等组织块，当 OB86 执行时可以通过它的临时变量读出产生故障的错误代码和事件类型，通过它们的组合可以得出具体的错误信息，这些信息可以通过 OB86 的在线帮助查到，同时也可以读到产生错误的模块地址和机架的信息，临时变量的具体含义请参阅在线帮助。STEP7 不能时时监控程序的运行，可用 Variable Table 监控实时数据变化。具体程序参见 OB_Example/OB86_Example。

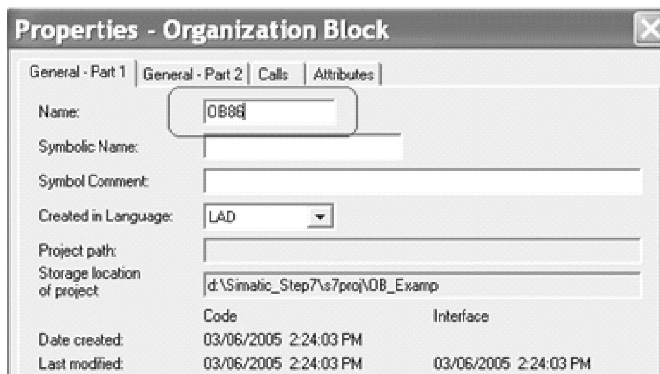


图 6-40　插入组织块 OB86

在 OB86_Example 程序的 Blocks 中插入组织块 OB86，如图 6-40 所示。

然后打开组织块 OB86 编写程序，OB86 的 STL 程序（可转成梯形图）为：

NetWork1：

A(

```
A(
A(
L    #OB86_EV_CLASS
T    MB    0
SET
SAVE
CLR
A    BR
)
JNB_001
L    #OB86_FLT_ID
T    MB    1
SET
SAVE
CLR
_001:    A    BR
)
JNB_002
L    #OB86_MDL_ADDR
T    MW    2
SET
SAVE
CLR
_002:    A    BR
)
JNB_003
L    #OB86_Z23
T    MD    4
_003:    NOP    0
```

注意：将 OB86 的临时变量 OB86_RACKS_FLTD Array [0..31] 改为 OB86_Z23 DWORD。将 OB86 和硬件组态下载到 CPU。在 OB86_Example 程序的 Blocks 中插入 Variable Table，然后打开，填入地址“MB0、MB1、MW2、MD4”并点击“Monitor Variable”按钮，设置画面如图 6-41 所示。

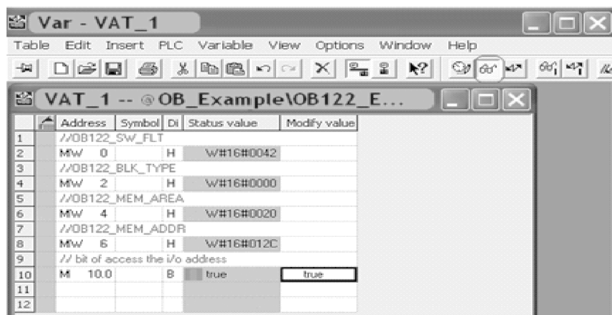


图 6-41 设置画面

此时可以读到 MB0、MB1 为 16#39 和 16#C4，可以通过它们的组合得出主站逻辑地址为 2047 的站有通信错误，出现错误的从站地址为 3。更多的信息读取请参阅 OB86 的在线帮助。

8. 起动的类型（OB100）

（1）硬件组态

在“OB_Example”项目中插入一S7300 站，命名为“OB100_Example”，然后插入“CPU 315-2DP”，参见 OB10 硬件组态。

（2）OB100 程序执行

OB100 程序在 CPU 执行 Warm Restart 时执行，且只执行一次，可用于变量的初始化，用 STEP 不能时时监控程序运行，可用 Variable Table 监控数据变化。具体程序参见 OB_Example/OB100_Example。在 OB100_Example 程序的 Blocks 中插入组织块 OB100，如图 6-42 所示。

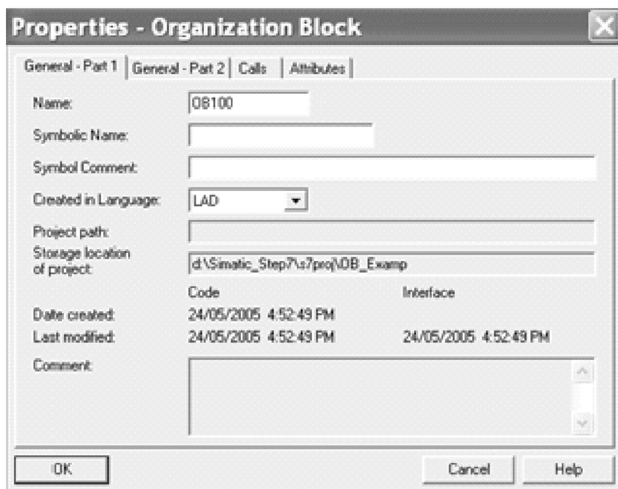


图 6-42　插入组织块 OB100

然后打开组织块 OB100 编写程序，OB100 的 STL 程序（可转成梯形图）为：

```
NetWork1:
L   123
T   MW   0
NOP   0
```

在 OB100_Example 程序的 Blocks 中插入 Variable Table，然后打开，填入 地址 MW0 并点击“Monitor Variable”按钮，设置画面如图 6-43 所示。

此时可以监控 MW0 为 123，如果修改 MW0 的值为 0，则 MW0 不会再被赋值为 123，只有当 CPU 再次执行 Warm Restart（重新上电或者从 Stop 切换到 Run 状态）后才会被赋值。

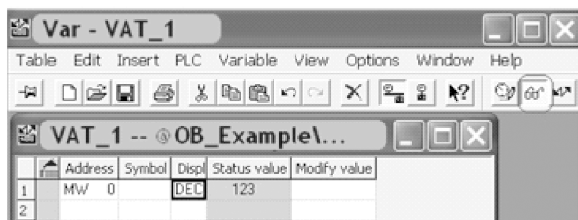


图 6-43　设置画面

9. 编程故障组织块（OB121）

（1）硬件组态

在“OB_Example”项目中插入一S7300 站，命名为“OB121_Example”，然后插入“CPU 315-2DP”，参见 OB10 硬件组态。

（2）OB121 程序执行

OB121 程序在 CPU 程序执行错误时执行，此错误不包括用户程序的逻辑错误和功能错误等，例如当 CPU 调用一未下载到 CPU 中的程序块，CPU 会调用 OB121，通过临时变量 OB121_BLK_TYPE 可以得出出现错误的程序块。使用 STEP7 不能时时监控程序的运行，可用 Variable Table 监控数据变化。具体程序参见 OB_Example/OB121_Example。

1）在 OB121_Example 程序的 Blocks 中插入组织块 OB121，然后打开组织块 OB121 编写程序。OB121 的 STL 程序（可转成梯形图）为：

```
NetWork1:
L   #OB121_BLK_TYPE
T   MW   0
```

NOP　0

2）在 OB121_Example 程序的 Blocks 中插入 FC1，然后打开 FC1 编写程序。FC1 的 STL 程序（可转成梯形图）为：

```
NetWork1：
A   #in1
 =   #out1
```

3）打开 OB1 编写程序，OB1 的 STL 程序（可转成梯形图）为：

```
NetWork1：
A   M   20.1
 =   L   20.0
BLD   103
A   M   10.0
JNB_001
CALL      FC   1
in1   ：=L20.0
out1：=M20.2
_001：   NOP   0
```

先将硬件组态和 OB1 下载到 CPU 中，此时 CPU 能正常运行。在 OB121_Example 程序的 Blocks 中插入 Variable Table，然后打开，填入地址 MW0 和 M10.0 并点击 “Monitor Variable” 按钮，程序运行正常，将 M10.0 置为 true，CPU 报错并停机，查看 CPU 的诊断缓冲区信息，发现为编程错误，将 OB121 下载到 CPU 中，再将 M10.0 置为 true，CPU 会报错误但不停机，MW0 立刻为 16#88，查看 OB121 的在线帮助，16#88 表示为 OB 程序错误，检查发现 FC1 未下载。设置画面如图6-44 所示。

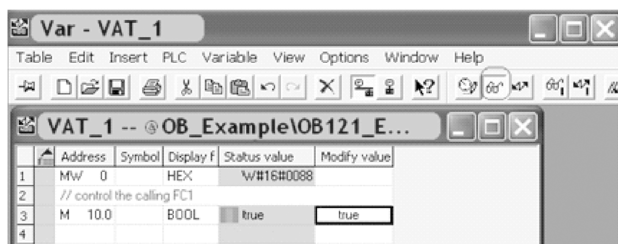


图 6-44　设置画面

下载 FC1 后再将 M10.0 置为 true，CPU 不会再报错，程序也不会再调用 OB121。

10. I/O 访问故障组织块（OB122）

（1）硬件组态

在 “OB_Example” 项目中插入一 S7300 站，命名为 “OB122_Example”，然后插入 “CPU315-2DP” 和一块模拟量输入模板 “6ES7 331-7KF02-0AB0”，配置 “SM331-7KF02-0AB0” 模块的 “Inputs” 选项，选择所有通道组为电压类型，注意模板的量程卡与设置的相同。模拟量的逻辑输入地址为 256...271，配置完成的画面如图 6-45 所示。

硬件组态完成后，保存编译。

（2）OB122 程序执行

OB122 程序在出现 I/O 访问错误时被调用，例如当 CPU 程序访问一未定义的 I/O 地址，CPU 会出现 I/O 访问错误，CPU 会调用 OB122，如果 OB122 未下载，CPU 会报故障停机。通过临时变量 OB122_SW_FLT 可以读出错误代码，通过 OB122_BLK_TYPE 得出出现错误的程序块，通过 OB122_MEM_AREA 可以读出被访问的地址类型，通过 OB122_MEM_ADDR 可以读出发生错误的

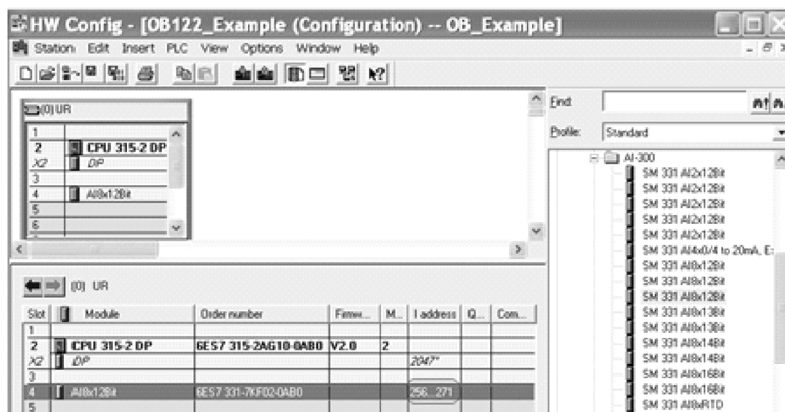


图 6-45 配置完成的画面

存储器地址。使用 STEP7 不能时时监控程序的运行，可用 Variable Table 监控数 据变化。具体程序参见 OB_Example/OB122_Example。

1）在 OB122_Example 程序的 Blocks 中插入组织块 OB122，然后打开组织块 OB122 编写程序，OB122 的 STL 程序（可转成梯形图）为：

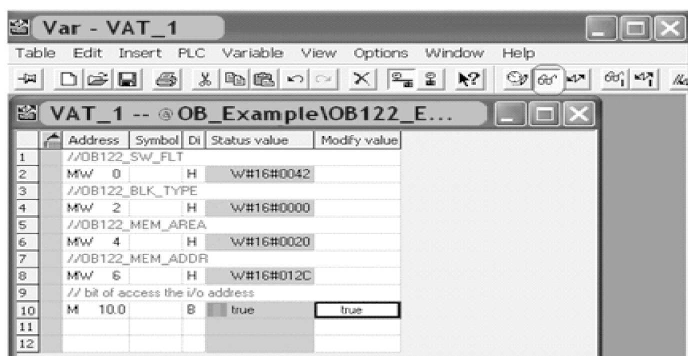
```
NetWork1：
A(
A(
A(
L   #OB122_SW_FLT
T   MW   0
SET
SAVE
CLR
A   BR
)
JNB_001
L   #OB122_BLK_TYPE
T   MW   2
SET
SAVE
CLR
_001：    A   BR
)
JNB_002
L   #OB122_MEM_AREA
T   MW   4
SET
SAVE
CLR
```

```
_002:    A   BR
)
JNB_003
L   #OB122_MEM_ADDR
T   MW   6
_003:    NOP   0
```

2）打开 OB1 编写程序，OB1 的 STL 程序（可转成梯形图）为：

```
NetWork1:
A   M   10.0
JNB_001
L   PIW   300
T   MW   20
_001:    NOP   0
```

先将硬件组态和 OB1 下载到 CPU 中，此时 CPU 能正常运行，在 OB122_Example 程序的 Blocks 中插入 Variable Table，然后打开，填入地址 MW0、MW2、MW4、MW6 和 M10.0 并点击“Monitor Variable”按钮，程序运行正常，将 M10.0 置为 true，CPU 会报错误并停机。查看 CPU 的诊断缓冲区信息，发现为 I/O 访问错误，将 OB122 下载到 CPU 中，再将 M10.0 置为 true，CPU 会报错误但不停机，MW0 为 16#0042，MW2 为 16#0000，MW4 为 16#00200，MW62 为 16#012C，查看 OB121 的在线帮助可得到相应的故障信息，具体监控画面如图 6-46 所示。



	Address	Symbol	Di	Status value	Modify value
1	//OB122_SW_FLT				
2	MW 0		H	W#16#0042	
3	//OB122_BLK_TYPE				
4	MW 2		H	W#16#0000	
5	//OB122_MEM_AREA				
6	MW 4		H	W#16#0020	
7	//OB122_MEM_ADDR				
8	MW 6		H	W#16#012C	
9	// bit of access the i/o address				
10	M 10.0		B	true	true
11					
12					

图 6-46 监控画面

检查并修改 OB1 程序为：

```
NetWork1:
A   M   10.0
JNB_001
L   PIW   256
T   MW   20
_001:    NOP   0
```

重新下载 OB1，运行程序 CPU 不会再报错，程序能正常运行。

11. 起动的类型（OB101）

（1）硬件组态

在“OB_Example”项目中插入一 S7400 站，命名为“OB101_Example”，然后插入“CPU 412-1（6ES7 412-1XF03-0AB0 Ver1.2）”，组态完成画面如图 6-47 所示。

双击“CPU 412-1”，设置起动方式，选择“Hot Restart”，具体画面如图 6-48 所示。

组态完成后保存编译。

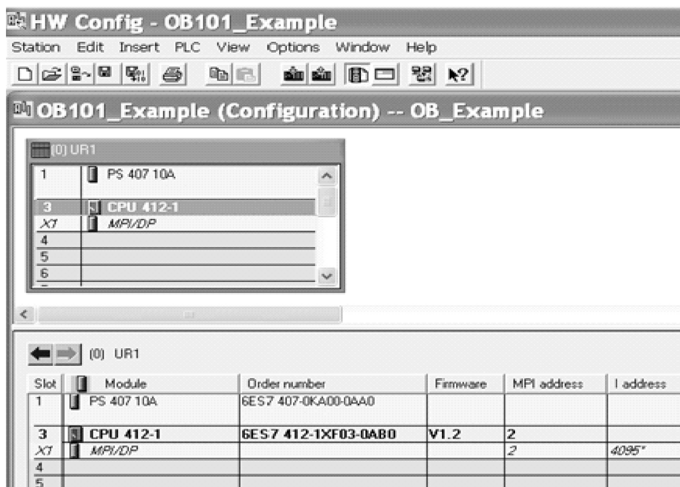


图 6-47　组态完成画面

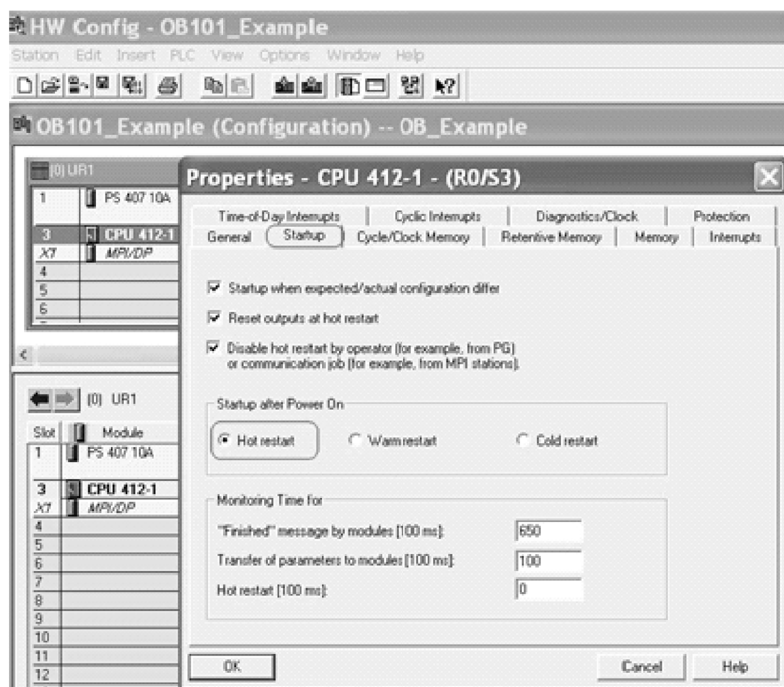


图 6-48　具体画面

（2）OB101 程序执行

OB101 程序在 CPU 执行 Hot Restart 时执行且只执行一次，可用于变量的初始化，使用 STEP7 不能时时监控程序的运行，可用 Variable Table 监控数据变化。具体程序参见 OB_Example/OB101_Example。在 OB101_Example 程序的 Blocks 中插入组织块 OB101，然后打开组织块 OB101 编写程序，OB101 的 STL 程序（可转成梯形图）为：

```
NetWork1：
L   123
T   MW   0
NOP   0
```

将程序和硬件组态下载到 CPU 中，然后执行 Hot Restart。

在 OB101_Example 程序的 Blocks 中插入 Variable Table，然后打开，填入 地址 MW0 并点击"Monitor Variable"按钮，设置画面如图 6-49 所示。

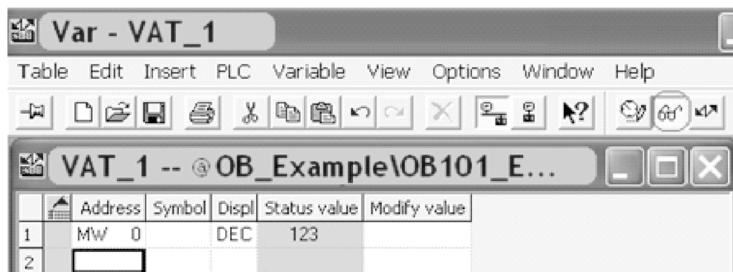


图 6-49 设置画面

此时可以监控 MW0 为 123，如果修改 MW0 的值为 0，则 MW0 不会再被赋值为 123，只有当 CPU 再次执行 Hot Restart 后才会被赋值。

12. 起动的类型（OB102）

（1）硬件组态

在"OB_Example"项目中插入一 S7400 站，命名为"OB102_Example"，然后插入"CPU 412-1（6ES7 412-1XF03-0AB0 Ver1.2）"，组态参见 OB101 部分，设置起动方式，选择"Cold Restart"，具体画面如图 6-50 所示。

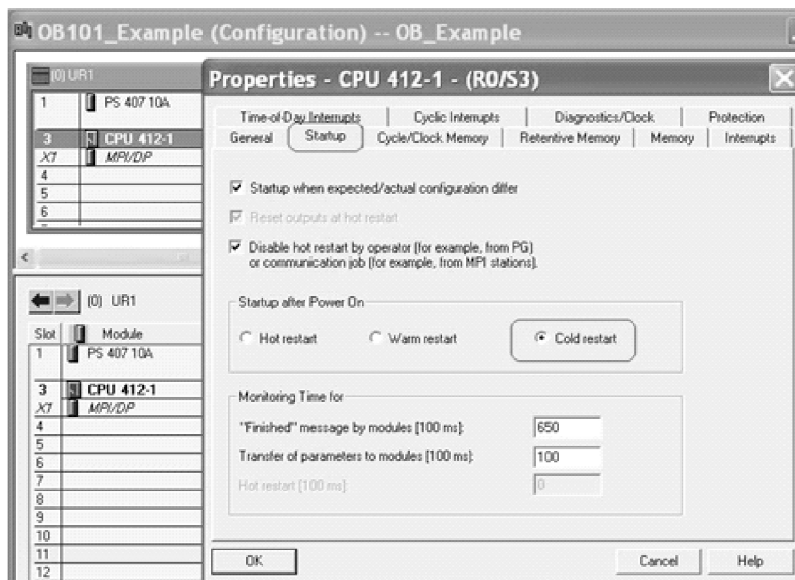


图 6-50 硬件组态画面

组态完成后保存编译。

（2）OB102 程序执行

OB102 程序在 CPU 执行 Cold Restart 时执行，且只执行一次，可用于变量 的初始化，使用 STEP7 不能时时监控程序的运行，可用 Variable Table 监控数据变化。具体程序参见 OB_Example/OB102_Example。在 OB102_Example 程序的 Blocks 中插入组织块 OB102，然后打开组织块 OB102 编写程序，OB102 的 STL 程序（可转成梯形图）为：

```
NetWork1:
L    123
T    MW    0
NOP    0
```

将程序和硬件组态下载到 CPU 中，然后执行 Cold Restart。在 OB102_Example 程序的 Blocks 中插入 Variable Table，然后打开，填入地址 MW0 并点击 “Monitor Variable” 按钮，设置画面如图 6-51 所示。

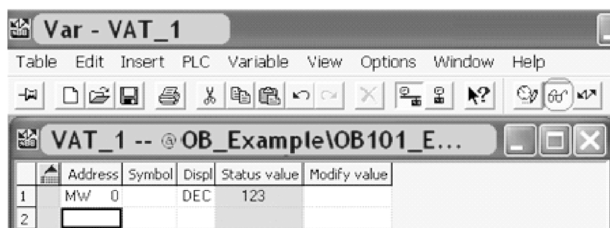


图 6-51　设置画面

此时可以监控 MW0 为 123，如果修改 MW0 的值为 0，则 MW0 不会再被赋值为 123，只有当 CPU 再次执行 Hot Restart 后才会被赋值。

13. 有关组织块的常问问题

(1) CPU 的 SF 红灯亮，CPU 停机是什么原因造成的?

当 CPU 的 SF 红灯亮，CPU 停机后不知道是什么原因，此时怎么办呢? 您需要去查看 CPU 的诊断缓冲区，根据缓冲区中提供的停机信息采取相应的措施，例如需要 OB82、OB86 的组织块下载等。那么如何查看 CPU 的诊断缓冲区呢?

方法一：首先建立 STEP7 与 CPU 的通信，然后打开硬件组态，点击 “Offline <- > Online” 按钮，然后双击 CPU，选择 “Diagnostic Buffer” 选项，可以查看 CPU 的故障信息，具体画面如图 6-52、图 6-53 所示。

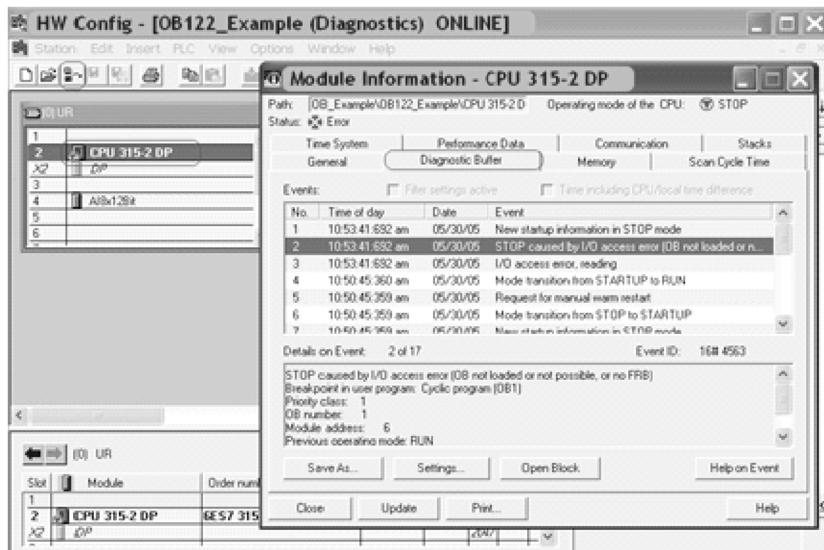


图 6-52　设置画面（一）

方法二：首先建立 STEP7 与 CPU 的通信，然后打开硬件组态，点击 CPU，然后选择 “PLC→Module Information...” 选项，具体画面如图 6-54 所示。

再选择 “Diagnostic Buffer” 选项，可以查看 CPU 的故障信息，具体画面同上。

(2) 为什么监控 OB100 程序时，感觉程序没有运行?

因为 OB100 为暖起动组织块，只有当 CPU 执行暖起动操作时才执行 OB100 的程序，且只执

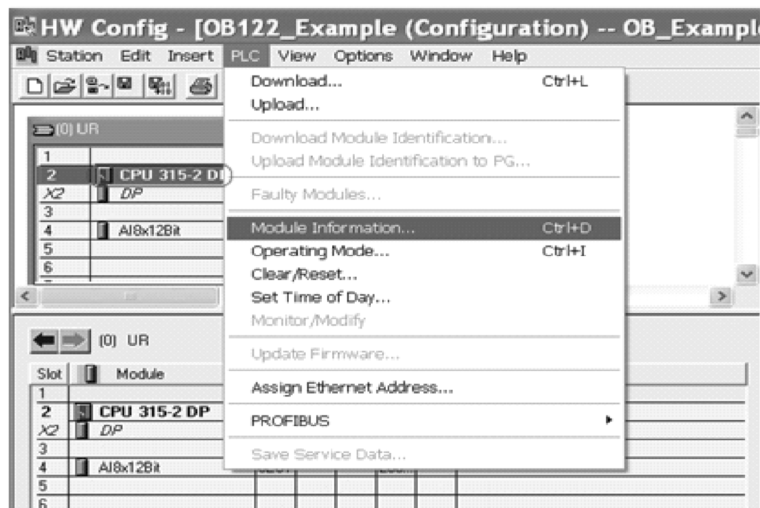


图 6-53　设置画面（二）

行一个周期，所以当监控 OB100 程序时感觉程序没有运行。

（3）OB35 的循环时间最长为 60s，但想实现 5min 的循环周期怎么办？

将 OB35 的执行周期设为 60000ms，在组织块 OB35 中做一加法计数，当计数值等于 5 后执行相应的程序，然后将计数值清零，简单程序举例如下。组织块 OB35 的 STL 程序为：

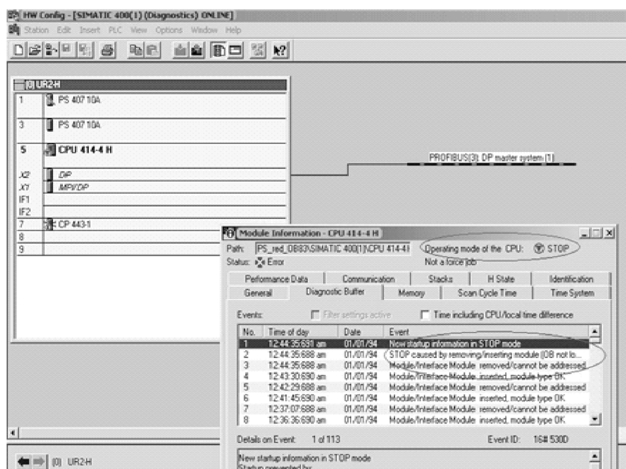


图 6-54　设置画面

```
NetWork1：
L   MW   0
L   1
 +I
T   MW   0
NOP   0
NetWork2：
L   MW   0
L   5
 = =I
 =    L   20.0
A   L   20.0
JNB_001
L   1234
T   MW   100
_001：   NOP   0
A   L   20.0
```

```
JNB_002
L   0
T   MW   0
_002：   NOP   0
```

（4）在冗余电源配置中，电源模块掉电，调用哪个 OB 可以防止 CPU 停机？

通过在程序中添加 OB83 可以防止 CPU 停机，而添加 OB81 不能防止 CPU 停机。通常我们很容易以为 OB81 就是处理所有电源故障的 OB，但对于冗余电源配置中，某个电源模块掉电故障，实际上 CPU 当作模块插拔故障来处理，因此需调用 OB83。如图 6-55 所示当程序中没有插入 OB83 时电源模块掉电，CPU 会停机。查看“Diagnostic Buffer”中显示的信息是模块插拔故障导致停机。

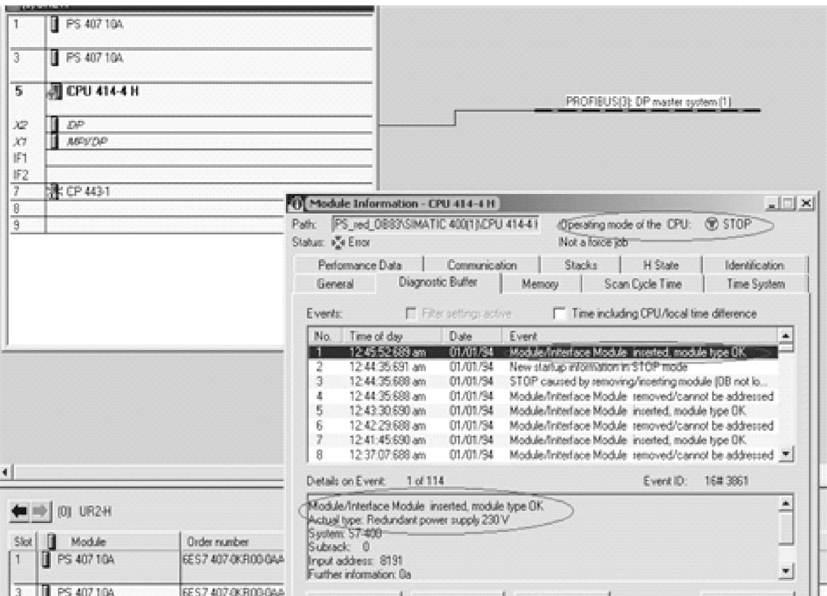


图 6-55　查看故障信息（一）

如图 6-56 所示当程序中没有插入 OB83 时电源模块掉电后恢复，CPU 停机不 恢复。查看“Diagnostic Buffer”中显示的信息是模块插入恢复。

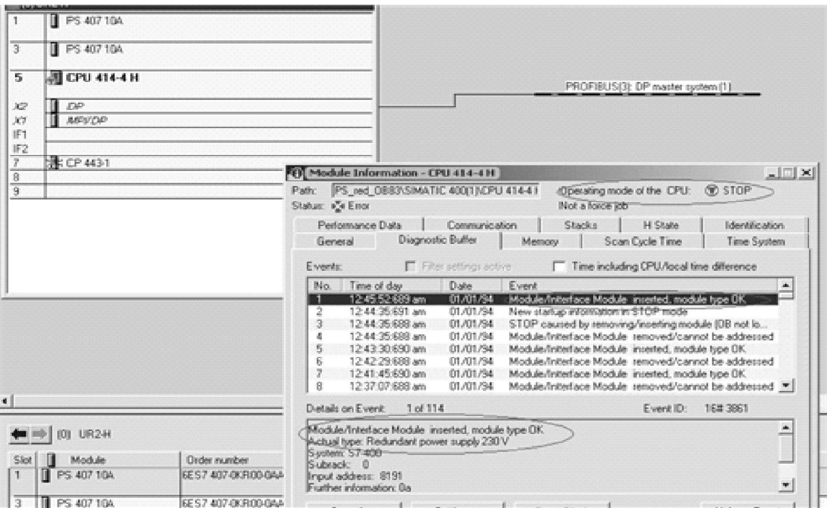


图 6-56　查看故障信息（二）

如图 6-57 所示当程序中插入 OB83 时电源模块掉电，CPU 不会停机。查看“Diagnostic Buffer”中显示的信息是模块拔除故障调用 OB83。

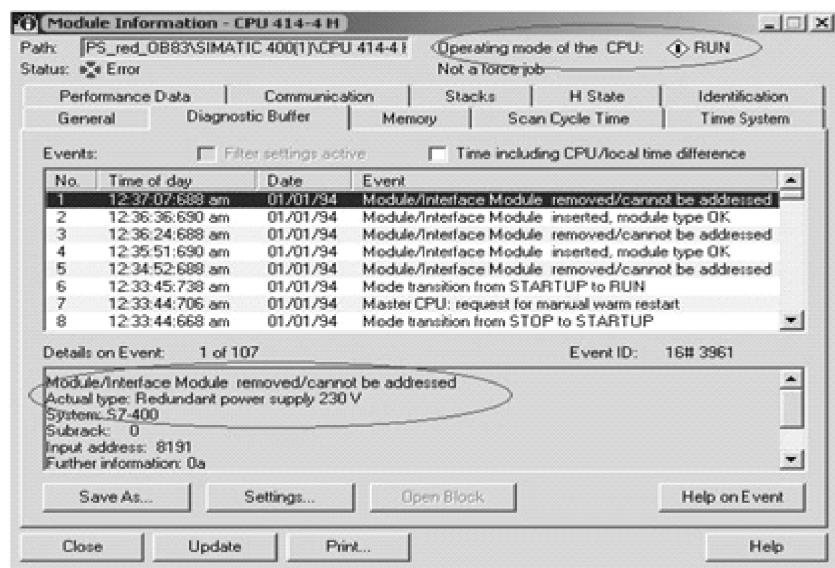


图 6-57　查看故障信息（三）

如图 6-58 所示当程序中插入 OB83 时电源模块掉电后恢复，CPU 不停机，外部故障灯恢复。查看“Diagnostic Buffer”中显示的信息是模块插入恢复。

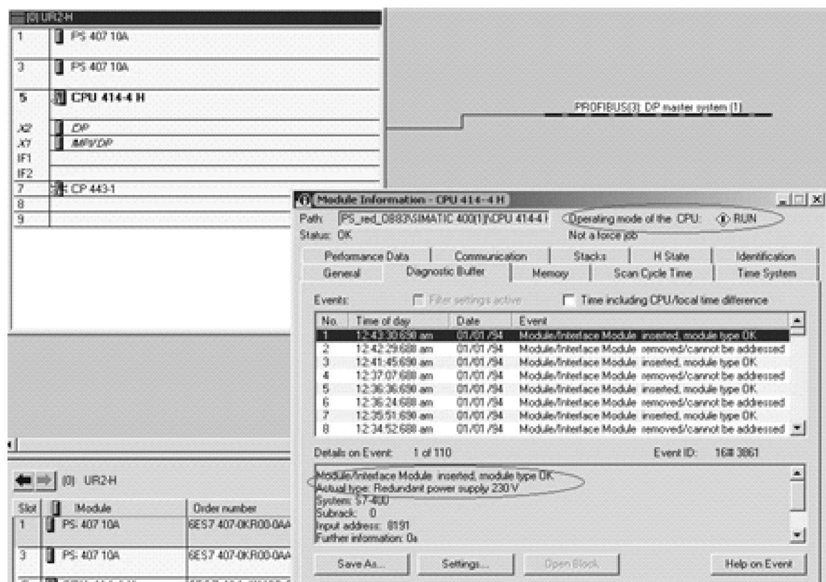


图 6-58　查看故障信息（四）

第六节　常用模拟量的处理

一、模拟量模块的用途

1. 原理

在生产过程中，存在大量的物理量，如压力、温度、速度、旋转速度、pH 值、粘度等。为

了实现自动控制，这些模拟信号需要被 PLC 处理。

2. 传感器

测量传感器利用线性膨胀、角度扭转或电导率变化等原理来测量物理量的变化。

3. 变送器

测量变送器将传感器检测到的变化量转换为标准的模拟信号，如：±500mV，±10V，±20mA，4～20mA。这些标准的模拟信号将接到模拟输入模块上。

4. 模数转换器

必须把模拟值转换为数字量，才能被 CPU 处理。模拟输入模块中的 ADC（模数转换器）用来实现转换功能。模数转换是顺序执行的，也就是说每个模拟通道上的输入信号是轮流被转换的。

5. 结果存储器

模数转换的结果存在结果存储器中，并一直保持到被一个新的转换值所覆盖。可用“L PIW…”指令来访问模数转换的结果。

6. 模拟输出

传递指令“T PQW…”用来向模拟输出模块中写模拟量的数值（由用户程序计算所得），该数值由模块中的 DAC（数模转换器）变换为标准的模拟信号。

7. 模拟执行器

采用标准模拟输入信号的模拟执行器可以直接连接到模拟输出模块上。

模拟量模块的用途如图 6-59 所示。

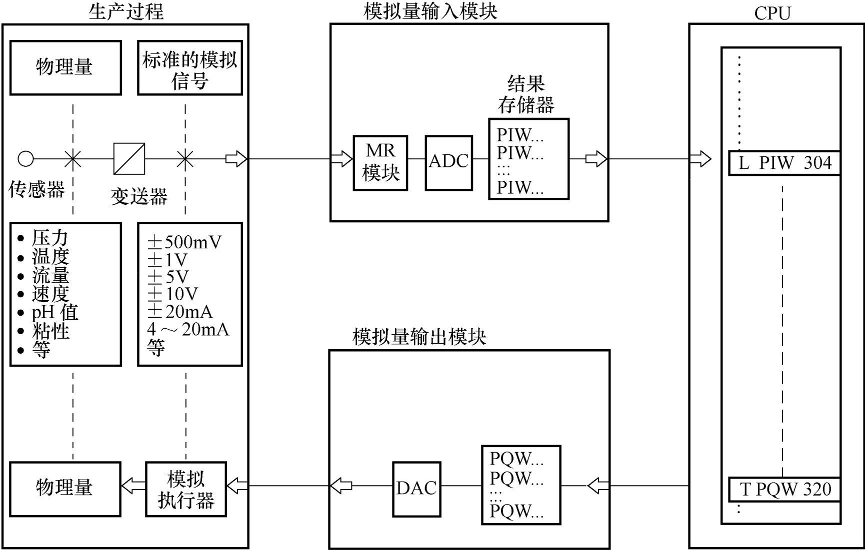


图 6-59　模拟量模块的用途

8. 测量的类型

通过量程卡上的适配开关可以设定测量的类型和范围。

没有量程卡的模拟量模块具有适应电压和电流测量的不同接线端子，这样，通过正确地连接有关端子可以设置测量的类型。

9. 量程卡

具有适配开关的量程卡安放在模块的左侧。在安装模块前必须正确地设置它。

允许的设置为“A”，“B”，“C”和“D”。关于设置不同的测量类型及测量范围的简要说明印在量程卡上。量程卡如图 6-60 所示。

10. 通道组

在一些模块上，几个通道被组合在一起构成一个通道组。此时，适配开关的设定应用于整个通道组。

11. 地址范围

S7-300 为模拟量输入和输出保留了特定的地址区域，以便与数字模块的输入、输出映像区的地址（PII/PIQ）区分开。地址范围为字节 256 ~ 767，每个模拟量通道占 2 字节。

12. 访问

用装载和传送指令来访问模拟模块：例如：指令“L PIW256”读取机架 0 上第一个模块的 1 通道的值。对于 S7-400 模拟量模块的地址区域从字节 512 开始。S7-300 模拟量模块的寻址如图 6-61 所示。

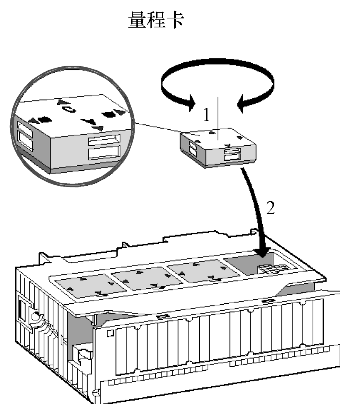


图 6-60　量程卡

Properties - AI2x12 bits - (R0/S10)
General　Addresses　Inputs
Inputs
Start: 352　　Part process image
End: 355　　No.: 0

机架 3	电源模块		IM（接收）	640 to 654	656 to 670	672 to 686	688 to 702	704 to 718	720 to 734	736 to 750	752 to 766
机架 2	电源模块		IM（接收）	512 to 526	528 to 542	544 to 558	560 to 574	576 to 590	592 to 606	607 to 622	624 to 638
机架 1	电源模块		IM（接收）	384 to 398	400 to 414	416 to 430	432 to 446	448 to 462	464 to 478	480 to 494	496 to 510
R 0	电源模块	CPU	IM（发送）	256 to 270	272 to 286	288 to 302	304 to 318	320 to 334	336 to 350	352 to 366	368 to 382
槽口号		2	3	4	5	6	7	8	9	10	11

图 6-61　S7-300 模拟量模块的寻址

二、模拟量寻址

在第一个信号模块插槽位置的模拟量输入/输出板的地址为 256。每个模拟量模块自动按 16 个字节的地址寄存器分配地址。每个模拟量值占用 2 个字节，所以，在用户程序中的模拟量地址应该使用偶数，以免使用数据错误。模拟量模块的输入/输出通道从实际插槽的相同基地址开始编号。S7-300 系统的实际 I/O 与 CPU 内的外设存储区（PI 和 PQ）相对应。S7-300/400 对模拟量有指定的寄存器，如 PII、PIQ，它们在每个扫描周期自动更新。相反地，在用户程序中，通过访问模拟量地址可以更新数据。模拟量输入的标识是 PIW，模拟量输出的标识是 PQW。因为模拟量的起始地址是 256，所以在第一个机架的第一个模块上，第一个通道的地址是 PIW256，最后一个模拟量的地址是 766。例如：要访问机架 2 的第一个模块的第二个通道，模拟量输入地址是 PIW514。模拟量模块 SM335（输入）如图 6-62 所示。

1. 诊断中断

当诊断中断被激活且一个硬件故障出现时，例如电源故障，诊断中断 OB 81 被触发。为此，必须在“Diagnostics”区域中选择要监视的输入。只有对 4 ~ 20mA 的输入通道才能检测断线故障。

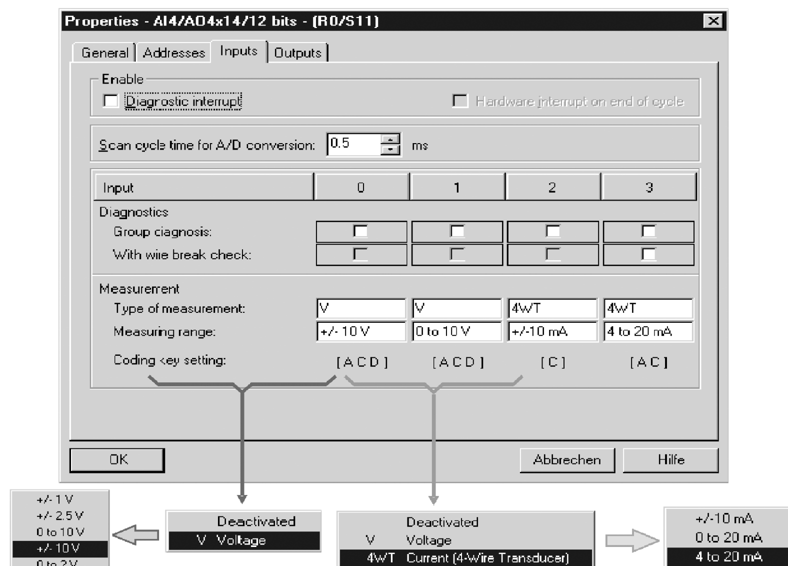


图 6-62　模拟量模块 SM335（输入）

2. 循检时间

循检时间是指模块对所有被激活的模拟输入都转换一次所需的时间。A-D 转换器的循检时间的允许设置范围是 0.5～16ms。当所有模拟输入都处理完毕后，模块可触发一次硬件中断（循检结束中断）（只有当设定的循检时间超过 1ms 时才有效）。

3. 说明

不使用的输入必须在硬件上短路连接，同时在软件上不激活（“deactivated”）。不激活的模拟输入可以减少循检时间。

4. 量程卡

当测量类型和范围都选择好之后，编码器的设置要求就显示出来。在本例中，测量卡必须插在“C”位置。

5. 分辨率

SM 335 输入信号的分辨率为 13 位 + 符号，模拟输出为 11 位 ＋ 符号。

模拟量模块 SM335（输出）如图 6-63 所示。

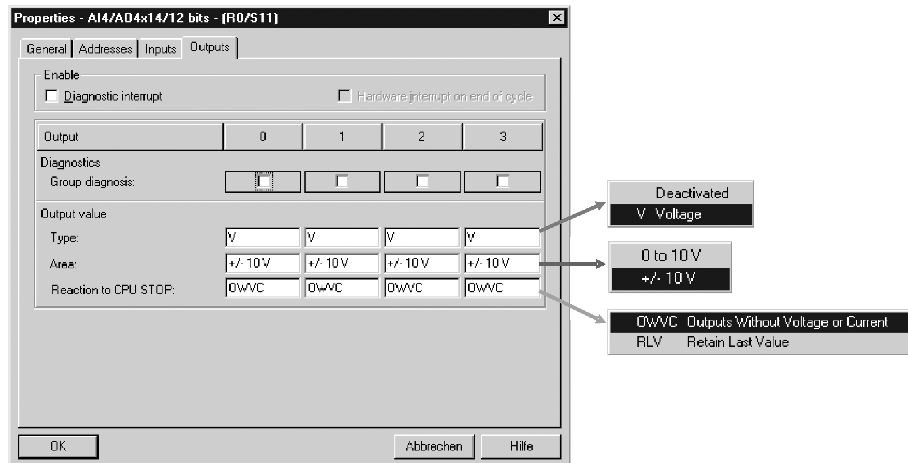


图 6-63　模拟量模块 SM335（输出）

6. 说明

不用的输出通道在硬件上必须保持开路（与模拟输入不同），在软件上不激活（“deactivated”）。

7. 参数

利用硬件组态工具，可以为该模块设置两组参数。

（1）总体的模块参数

- Diagnostic Interrupt（诊断中断）：如果“Group Diagnosis”检查框被激活后，有一个可诊断的事件发生，则有关信息被记录在模块的诊断数据区中并触发一个诊断中断 OB82。模拟量模板可以检测下列诊断事件：
- 参数分配/组态错误；
- 断线（如果“With Wire Break Check”检查框被激活）；
- 超出测量范围；
- 低于测量范围；
- 无负载电压 L+。
- Hardware Interrupt when Limit Value Exceeded（超界限硬件中断）：如果输入值超过用户定义的上界（“Upper Limit Value”）和下界（“Lower Limit Value”），则模块触发一个硬件中断。

注意：只有第一个通道具有监视输入超界限的功能！

（2）个体的输入参数

- Type of Measurement（测量类型）：单击该选项框，将显示可能的测量类型（电压、电流）。对于不使用的通道或通道组选择不激活（“deactivated”）选项，且必须将这些通道接到模块的机架地上。
- Measuring Range（测量范围）：单击该选项框，将显示在选定的测量类型下可能的测量范围。
- Coding Key Setting（编码器的设置）：当选好测量类型和测量范围后，正确地设置量程卡是十分必要的，此处显示量程卡的允许设置。
- integration time（积分时间）和 interference frequency suppression（干扰频率抑制）是相互依赖的。

8. 表达方式

模拟量用两种互补的形式表达：

位 15 =0 为正数，位 15 =1 为负数。

9. 分辨率

如果模拟量模块的分辨率小于 15 位，则模拟量写入累加器时向左对齐。不用的位用“0”填充。

10. 积分时间

分辨率是通过在硬件组态中选择积分时间来间接定义的。下表给出了积分时间、分辨率和扰动频率抑制三者的关系（对于 SM331 模块）：

积分时间（ms）	分辨率（位）	扰动频率抑制（Hz）
2.5	9+ 符号位	400
16.6	12+ 符号位	60
20	12+ 符号位	50
100	14+ 符号位	10

11. 精度

根据模块的种类，8～15 位的分辨率都是可能的。

12. 转换时间

转换时间取决于模块上模数转换方式（积分方式或连续逼近）。

在 S7-300 手册中可以查到各种模块的转换时间。例如：SM344 所有 4 个通道的转换时间都是 5ms。

模拟量的表达方式和测量值的分辨率见表 6-12。

表 6-12 模拟量的表达方式和测量值的分辨率 * =0 或 1

位的序号		单位		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
位值		十进制	16 进制	VZ	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
位的分辨率 + 符号	8	128	80	*	*	*	*	*	*	*	*	1	0	0	0	0	0	0	0
	9	64	40	*	*	*	*	*	*	*	*	*	1	0	0	0	0	0	0
	10	32	20	*	*	*	*	*	*	*	*	*	*	1	0	0	0	0	0
	11	16	10	*	*	*	*	*	*	*	*	*	*	*	1	0	0	0	0
	12	8	8	*	*	*	*	*	*	*	*	*	*	*	*	1	0	0	0
	13	4	4	*	*	*	*	*	*	*	*	*	*	*	*	*	1	0	0
	14	2	2	*	*	*	*	*	*	*	*	*	*	*	*	*	*	1	0
	15	1	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	1

13. 电压，电流

可设置的对称的电压或电流的范围：

(对称的) • ±80mV • ±2.5V • ±3.2mA

• ±250mV • ±5V • ±10mA

• ±500mV • ±10V • ±20mA

• ±1V

转换结果的额定范围为 −27648 ~ +27648。

可设置的不对称的电压或电流的范围：

(不对称的) • 0 ~ 2V • 0 ~ 20mA

• 1 ~ 5V • 4 ~ 20mA

转换结果的额定范围为 0 ~ +27648。

14. 电阻

可设置的电阻的范围：

• 0 ~ 150Ω

• 0 ~ 300Ω

• 0 ~ 600Ω

转换结果的额定范围为 0 ~ +27648。

15. 温度

温度用热电阻或热电偶来测量。转换结果的额定值用温度的十倍值来表示：

传感器：	温度范围：	转换结果的额定范围：
• Pt 100	−200 ~ +850℃	−2000 ~ +8500
• Ni 100	−60 ~ +250℃	−600 ~ +2500
• K 型热电偶	−270 ~ +1372℃	−2700 ~ +13720
• N 型热电偶	−270 ~ +1300℃	−2700 ~ +13000
• J 型热电偶	−210 ~ +1200℃	−2100 ~ +12000
• E 型热电偶	−270 ~ +1000℃	−2700 ~ +10000。

在不同测量范围下模拟量的表达方式见表 6-13。

表 6-13 在不同测量范围下模拟量的表达方式

范围	电压 例如：		电流 例如：		电阻 例如：		温度 例如 Pt100	
	测量范围 ±10V	单位	测量范围 4～20mA	单位	测量范围 0～300Ω	单位	测量范围 -200～+850℃	单位
超上限	> =11.759	32767	> =22.815	32767	> =352.778	32767	> =1000.1	32767
超上界	11.7589 : 10.0004	32511 : 27649	22.810 : 20.0005	32511 : 27649	352.767 : 300.011	32511 : 27649	1000.0 : 850.1	10000 : 8501
额定范围	10.00 7.50 : -7.5 -10.00	27648 20736 : -20736 -27648	20.000 16.000 : : 4.000	27648 20736 : : 0	300.000 225.000 : : 0.000	27648 20736 : : 0	850.0 : : : -200.0	8500 : : : -2000
超下界	-10.0004 : -11.759	-27649 : -32512	3.9995 : 1.1852	-1 : -4864	不允许负值	-1 : -4864	-200.1 : -243.0	-2001 : -2430
超下限	< = -11.76	-32768	< = -1.1845	-32768		-32768	< = -243.1	-32768

三、模拟输入量的规范化

示例：水罐的液位以升为单位来测量，它的容量是 500L。

例 A 显示当水罐空时传感器检测到 0V 电压，当水罐满时传感器检测到 10V 电压。

例 B 显示当水罐空时传感器检测到-10V 电压，当水罐满时传感器检测到 10V 电压。

分辨率：在例 B 中，测量的液位具有双倍的分辨率即可测量的变化的一半，因为水罐的容量被分配到 -27648 ~ +27648 的范围单位。

规范化：模拟模板 -10 ~ +10V 的电压范围对应 -27648 ~ +27648 数值范围。这个数值范围转化为实际物理范围称为规范化（或整定）。

标准块 FC 105 用于模拟量的规范化。在 STEP 7 软件的“Standard Library”库中的“TI-S7 Converting Blocks”S7 程序中提供了 FC 105。

IN：IN 输入端的模拟值可直接从模板上读取或从一个 INT 格式的数据接口上读取。

LO_LIM、HI_LIM、LO_LIM（下界）和 HI_LIM（上界）输入参数用于定义规范化的物理量范围。本例中，转换为 0 ~ 500L 范围。

OUT：规范化后的值（实际物理量）以实数格式存储在 OUT 输出端（LO_LIM < = OUT < = HI_LIM）。

BIPOLAR：BIPOLAR 输入端用来决定是否仅正值或负值也被转换。如果带有状态“0”（单向）的操作数被传送到该参数，做从 0 ~ +27648 范围的规范化。如果带有状态“1”（双向）的操作数被传送到该参数，做从 -27648 ~ +27648 范围的规范化。

RET_VAL：如果该程序块执行无误，则 RET_VAL 端输出为 0。

模拟输入量的规范化如图 6-64 所示。

FC105 的编程实现公式：

OUT = [(FLOAT(IN)/K) *(HI_LIM-LO_LIM)] + LO_LIM

其中，IN 为某一 AI 模块的一个通道值，例如 PIW256，WORD 类型

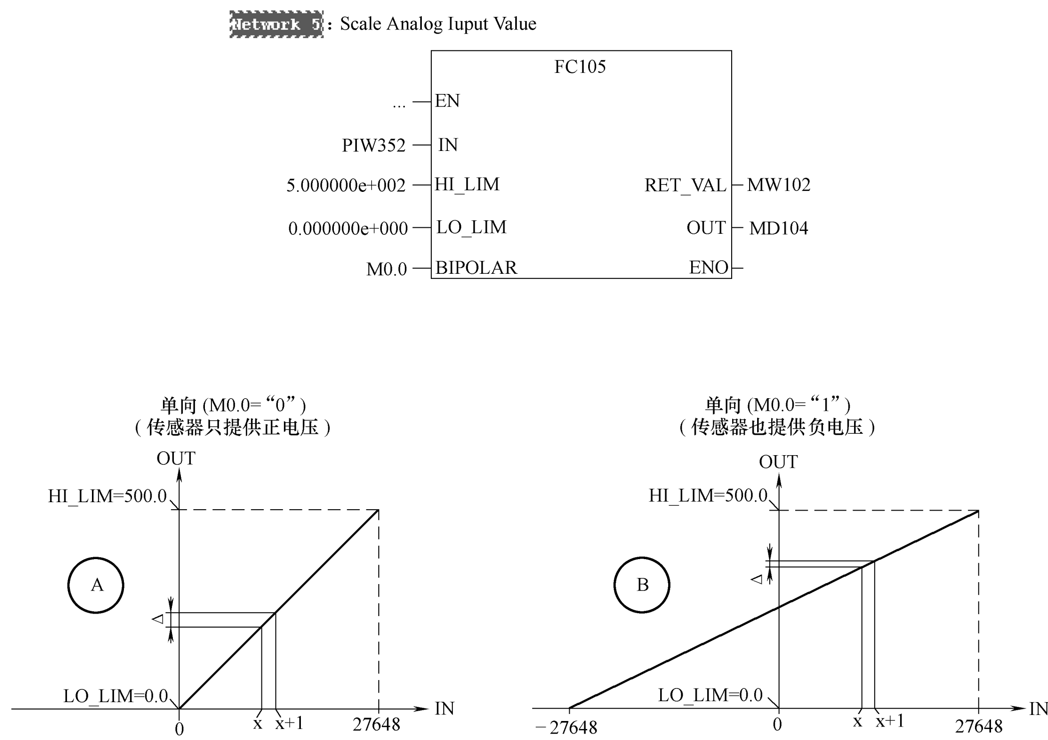


图 6-64　模拟输入量的规范化

$$y = [(FLOAT(IN)/27648.0) * (max - min)] + min$$

四、模拟量输出的规范化

示例：用户程序计算出的 0.0～100.0% 范围内的模拟量用 FC106 转化为范围 0～27648（单向）或 −27648～+27648（双向）。当转化的值输出到模拟输出模板，该模板将用例如 0～+10V（单向）或 −10～+10V（双向）的值驱动模拟量执行器（例如伺服阀）。

例 A 显示当程序值为 0% 时用 0 值（0V 或 0mA）驱动模拟量执行器，当程序值为 100% 时用最大值（例如 +10V 或 20mA）驱动模拟量执行器。

例 B 显示当程序值为 0% 时用最小值（−10V 或 −20mA）驱动模拟量执行器，当程序值为 100% 时用最大值（例如 +10V 或 20mA）驱动模拟量执行器。

规范化：程序计算出的值 – 示例中的百分比 – 必须转化为模拟输出模板的数值范围。

标准块 FC106 用于模拟输出操作规范化。在 STEP 7 软件的“Standard Library”库中的“TI-S7 Converting Blocks”S7 程序中提供了 FC 106。

IN：程序计算出的值必须以 REAL 格式传送。

LO_LIM、HI_LIM：LO_LIM（下界）和 HI_LIM（上界）输入参数用于定义程序值的范围。本例中，范围为 0.0%～100.0%。

OUT：规范化后的值以 INT 格式在 OUT 输出端输出。

BIPOLAR：BIPOLAR 输入端用来决定是否仅正值或负值也被转换。如果带有状态“0”（单向）的操作数被传送到该参数，做从 0～+27648 范围的规范化。如果带有状态“1”（双向）的操作数被传送到该参数，做从 −27648～+27648 范围的规范化。

RET_VAL：如果该程序块执行无误，则 RET_VAL 端输出为 0。

模拟量输出的规范化如图 6-65 所示。

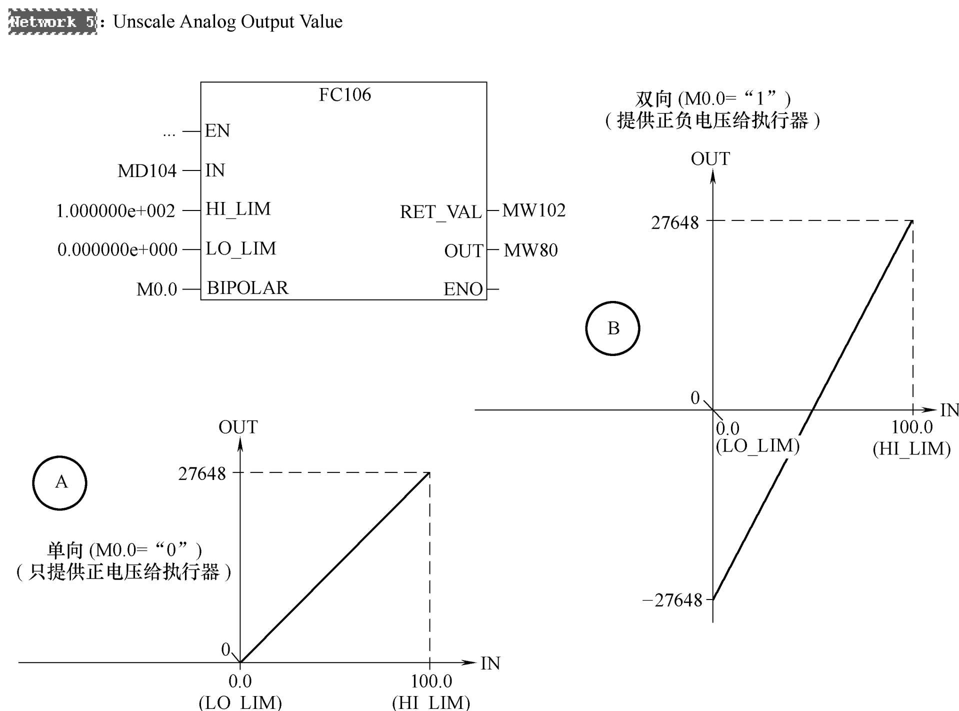


图 6-65 模拟量输出的规范化

FC106 接收一个以工程单位表示、且标定于下限和上限（LO_LIM 和 HI_LIM）之间的实型输入值（IN），并将其转换为一个整型值。将结果写入 OUT。

FC106 使用以下等式：

$$OUT = ((IN-LO_LIM)/(HI_LIM-LO_LIM)) * K1$$

模拟输出量的表达形式

电压，电流 －27648 ~ ＋27648 可转换为对称的电压或电流的额定范围：

(对称的)

- ±10V
- ±20mA

电压，电流 0 ~ ＋27648 可转换为不对称的电压或电流的额定范围：

(不对称的)

- 0 ~ 10V
- 1 ~ 5V
- 0 ~ 20mA
- 4 ~ 20mA

超限：如果被转换的数值超限，模拟输出模块被禁止（0V，0mA）。

模拟输出量的表达形式见表 6-14。

表 6-14　模拟输出量的表达形式

范围	单位	电压 输出范围：0~10V	电压 输出范围：1~5V	电压 输出范围：±10V	电流 输出范围：0~20mA	电流 输出范围：4~20mA	电流 输出范围：±20mA
超上限	>=32767	0	0	0	0	0	0
超上界	32511 : 27649	11.7589 : 10.0004	5.8794 : 5.0002	11.7589 : 10.0004	23.515 : 20.0007	22.81 : 20.005	23.515 : 20.0007
额定范围	27648 : 0 : -6912 -6913 : : : -27648	10.0000 : 0 0	5.0000 : 1.0000 0.9999 0 0	10.0000 : 0 : : : : : : -10.0000	20.000 : 0 0	20.000 : 4.000 3.9995 0 0	20.000 : 0 : : : : : : -20.000
超下界	-27649 : -32512			-10.0004 : -11.7589			-20.007 : -23.515
超下限	<=-32513			0			0

第七节　在 STEP7 中实现 PID 控制

一、概述

本文中所讨论的功能块（SFB41/FB41，SFB42/FB42，SFB43/FB43）仅仅是使用于 S7 和 C7 的 CPU 中的循环中断程序中。该功能块定期计算所需要的数据，保存在指定的 DB 中（背景数据块）。允许多次调用该功能块。CONT_C 块与 PULSEGEN 块组合使用，可以获得一个带有比例执行机构脉冲输出的控制器（例如，加热和冷却装置）。

- SFB41/FB41（CONT_C），连续控制方式；
- SFB42/FB42（CONT_S），步进控制方式；
- SFB43/FB43（PULSEGEN），脉冲宽度调制器。

注意：SFB41/42/43 与 FB41/42/43 兼容，可以用于 CPU 313C、CPU 313C-2 DP/PTP 和 CPU 314C-2 DP/PTP 中。

1. 应用

借助于大量模块组成的控制器，可以完成带有 PID 算法的实际控制器。控制效率，即处理速度取决于所使用的 CPU 性能。对于给定的 CPU，必须在控制器的数量和控制器所需要的执行频率之间找到一个折中方案。连接的控制电路越快，所安装的控制器数量越少，则每个时间单位计算的数值就越多。对于控制过程的类型没有限制。较慢（温度、填料位等）以及较快的控制系统（流量、速度等）都可以控制。

2. 控制系统分析

在工程实际中，应用最为广泛的调节器控制规律为比例（P）、积分（I）、微分（D）控制，简称 PID 控制，又称 PID 调节。PID 控制器问世至今已有近 70 年历史，它以其结构简单、稳定性好、工作可靠、调整方便而成为工业控制的主要技术之一。当被控对象的结构和参数不能完全掌握，或得不到精确的数学模型，控制理论的其他技术难以采用时，系统控制器的结构和参数必须依靠经验和现场调试来确定，这时应用 PID 控制技术最为方便。即当我们不完全了解一个系统和被控对象，或不能通过有效的测量手段来获得系统参数时，最适合用 PID 控制技术。PID 控制，实际中也有 PI 和 PD 控制。PID 控制器就是根据系统的误差，利用比例、积分、微分计算出控制量进行控制的。

比例控制：比例控制是一种最简单的控制方式。其控制器的输出与输入误差信号成比例关系。当仅有比例控制时系统输出存在稳态误差（Steady-State Error）。

积分控制：在积分控制中，控制器的输出与输入误差信号的积分成正比关系。对一个自动控制系统，如果在进入稳态后存在稳态误差，则称这个控制系统是有稳态误差的或简称有差系统（System With Steady-State Error）。为了消除稳态误差，在控制器中必须引入“积分项”。积分项对误差取决于时间的积分，随着时间的增加，积分项会增大。这样，即便误差很小，积分项也会随着时间的增加而加大，它推动控制器的输出增大使稳态误差进一步减小，直到等于零。因此，比例 + 积分（PI）控制器，可以使系统在进入稳态后无稳态误差。

微分控制：在微分控制中，控制器的输出与输入误差信号的微分（即误差的变化率）成正比关系。自动控制系统在克服误差的调节过程中可能会出现振荡甚至失稳。其原因是由于存在较大惯性组件（环节）或滞后（delay）组件，具有抑制误差的作用，其变化总是落后于误差的变化。解决的办法是使抑制误差的作用的变化“超前”，即在误差接近零时，抑制误差的作用就应该是零。这就是说，在控制器中仅引入“比例”项往往是不够的，比例项的作用仅是放大误差的幅值，而目前需要增加的是“微分项”，它能预测误差变化的趋势，这样，具有比例 + 微分的控制器，就能够提前使抑制误差的控制作用等于零，甚至为负值，从而避免了被控量的严重超调。所以对有较大惯性或滞后的被控对象，比例 + 微分（PD）控制器能改善系统在调节过程中的动态特性。

PID 常用口诀：参数整定找最佳，从小到大顺序查，先是比例后积分，最后再把微分加，曲线振荡很频繁，比例度盘要放大，曲线漂浮绕大弯，比例度盘往小扳，曲线偏离回复慢，积分时间往下降，曲线波动周期长，积分时间再加长，曲线振荡频率快，先把微分降下来，动差大来波动慢，微分时间应加长，理想曲线两个波，前高后低 4 比 1，一看二调多分析，调节质量不会低。

PID 控制器参数的工程整定，各种调节系统中 P. I. D 参数经验数据参照如下：

温度：P = 20% ~ 60%，T = 180 ~ 600s，D = 3 ~ 180s；

压力：P = 30% ~ 70%，T = 24 ~ 180s；

液位：P = 20% ~ 80%，T = 60 ~ 300s；

流量：P = 40% ~ 100%，T = 6 ~ 60s。

控制系统的静态性能（增益）和动态性能（滞后、空载时间、积分常数等），都是设计系统控制器及其静态参数（P 操作）和动态参数（I、D 操作）的主要因素。

因此，熟练掌握控制系统的类型和特性非常重要。各控制器波形分别如图 6-66、图 6-67、图 6-68、图 6-69 所示。

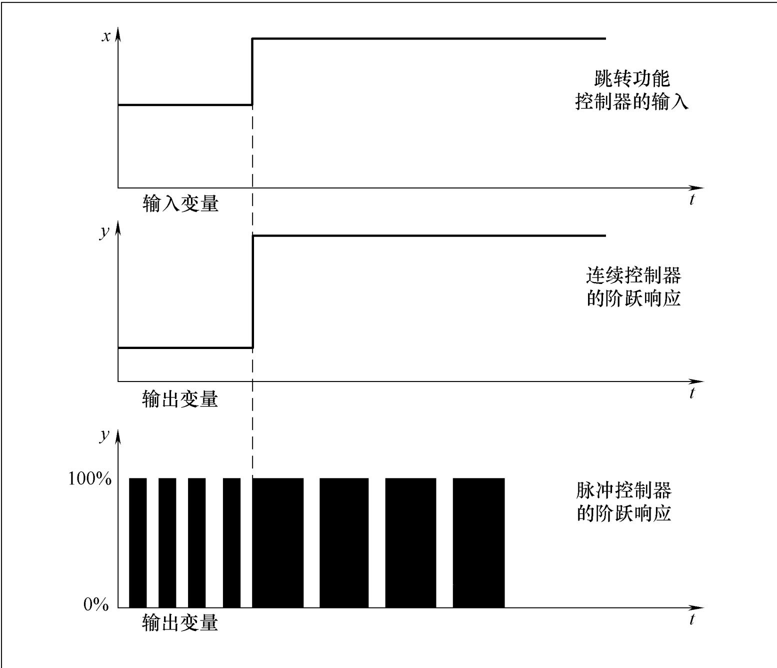


图 6-66　P 控制器波形

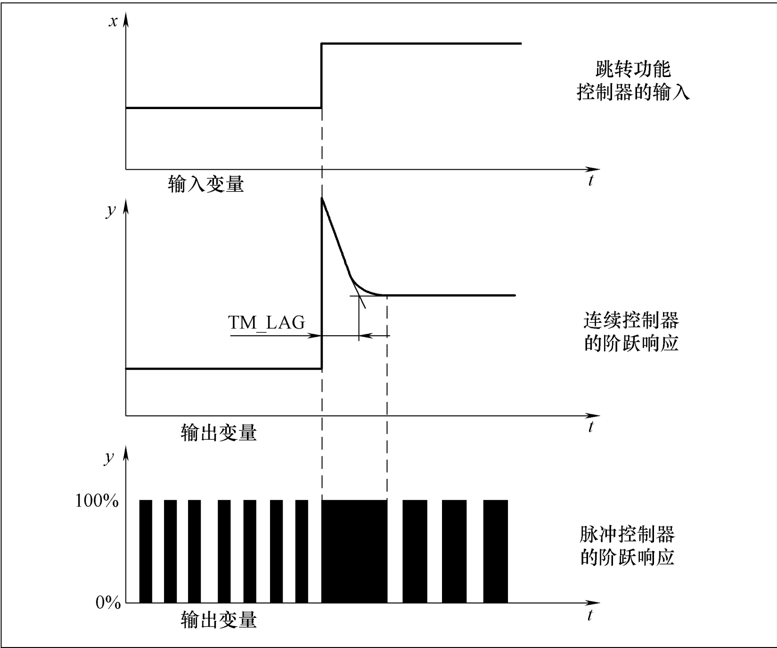


图 6-67　PD 控制器波形

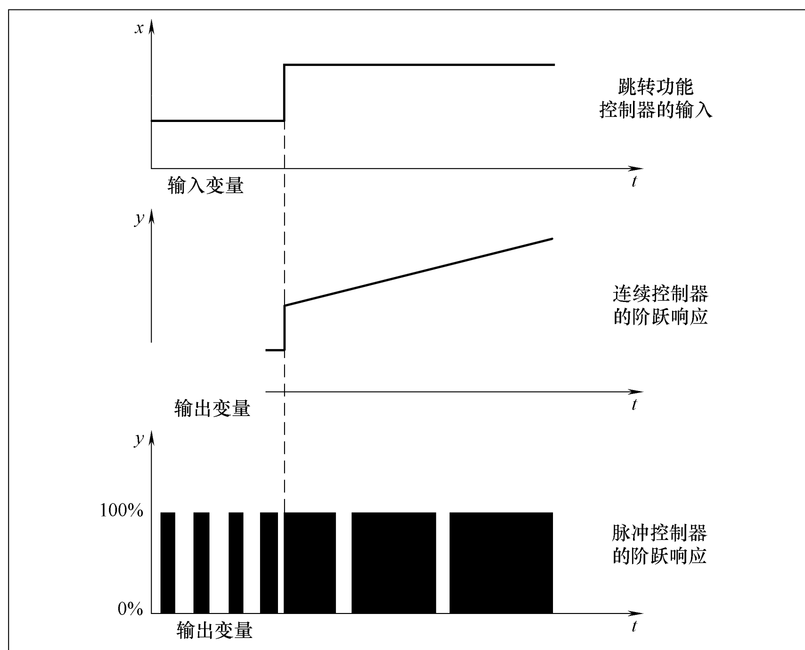


图 6-68 PI 控制器波形

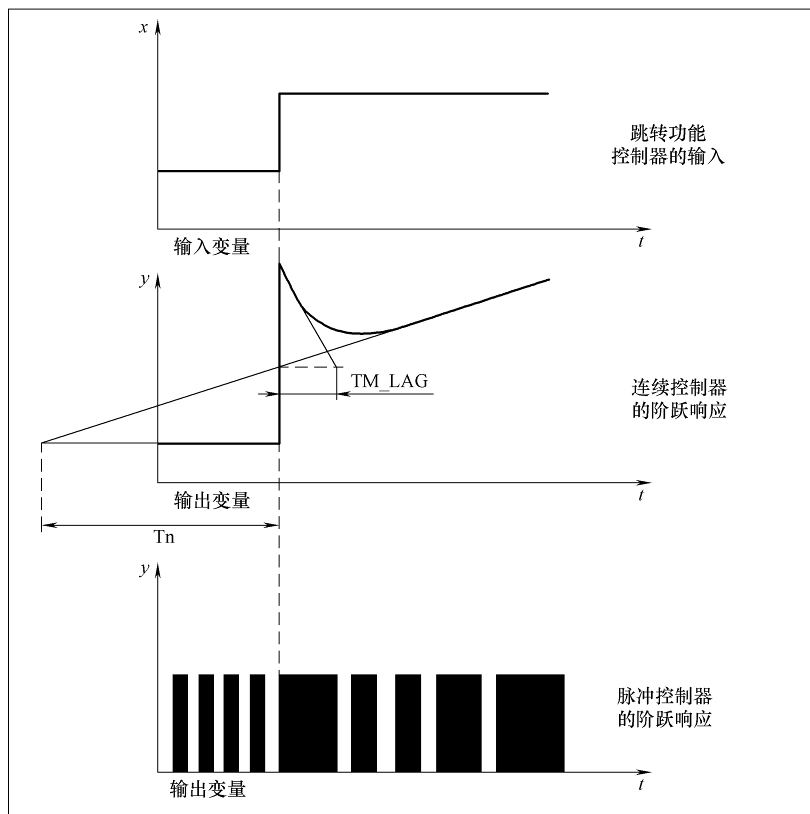


图 6-69 PID 控制器波形

二、PID 系统控制器的选择

控制系统的属性由技术过程和机器条件决定。因此，为了获得良好的控制效果，必须选择最适用的系统控制器。

1. 连续控制器、开关控制器

- 连续控制器，输出一个线性（模拟）数值。
- 开关控制器，输出一个二进制（数字）数值。

2. 固定值控制器

固定值控制，使用设定固定数值进行的过程控制，只是偶尔修改一下参考变量，过程偏差的控制。

3. 级联控制器

级联控制器，控制器串行连接控制。第一个控制器（主控制器）决定了串行控制器（从控制器）的设定点，或者根据过程变量的实际错误影响设定点。

一个级联控制器的控制性能可以使用其他的过程变量加以改进。为此，可以为主控制变量添加一个辅助过程变量 PV2（主控制器 SP2 的输出）。主控制器可以将过程变量 PV1 施加给设定点 SP1，并且可以调整 SP2，以便尽可能快地到达目标，而没有过调节。级联控制器控制性能的调节如图 6-70 所示。

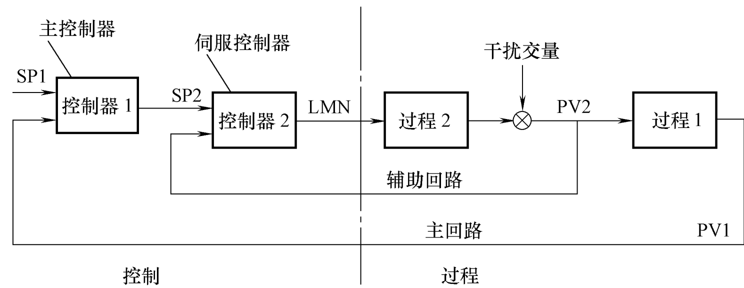


图 6-70　级联控制器控制性能的调节

4. 混合控制器

混合控制器是指根据每个被控组件所需要的设定点总数量，来计算总 SP 数量的一种控制结构。在此，混合系数 FAC 的和必须为“1”。混合控制过程如图 6-71 所示。

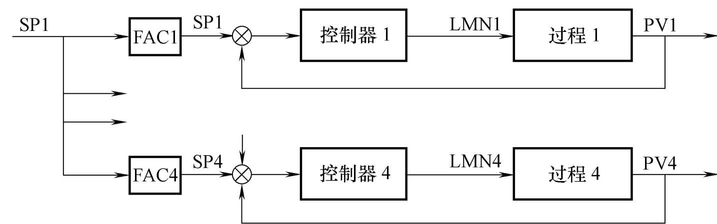


图 6-71　混合控制过程

5. 比例控制器

（1）单循环比例控制器

单循环比例控制器，可以用于“两个过程变量之间的比率”比“两个过程变量的绝对数值”重要的场合（例如，速度控制）。单循环比例控制过程如图 6-72 所示。

（2）多循环比例控制器

对于多循环比例控制，两个过程变量 PV1 和 PV2 之比保持为常数。因此，可以使用第一个

控制循环的过程数值，来计算第二个控制循环的设定点。对于过程变量 PV1 的动态变化，也可以保证保持特定的比例。多循环比例控制过程如图 6-73 所示。

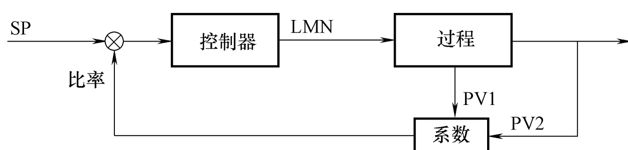


图 6-72　单循环比例控制过程

6. 二级控制器

一个二级控制器只能采集两个输出状态（例如，开和关）。典型的控制为：一个加热的系统，通过继电器输出的脉冲宽度调制。

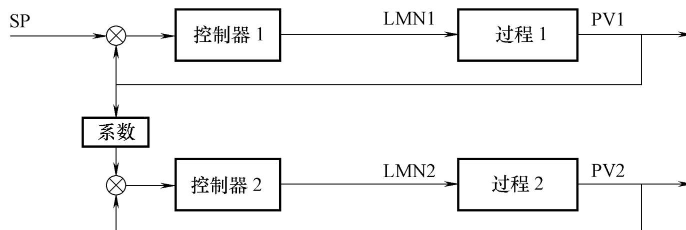


图 6-73　多循环比例控制过程

7. 三级控制器

一个三级控制器只能采集到 3 个具体的输出状态。需要区分：“脉冲宽度调制”（例如，加热-冷却，加热-关机-冷却）和“使用集成执行机构的步进控制”（例如，左-停止-右）之间的区别。

三、布线

对于没有集成的 I/O 控制器，必须使用附加的 I/O 模块。

1. 布线规则

（1）连接电缆

1）对于数字 I/O，如果线路有 100m 长，必须使用屏蔽电缆。

2）电缆屏蔽时必须在两端进行接地。

3）软电缆的截面积选择 0.25 ~ 1.5mm^2。

4）无需选择电缆套。如果决定选择使用电缆套，可以使用不带绝缘套圈的电缆套（DIN 46228，Shape A，Short version）。

（2）屏蔽端接元件

1）可以使用屏蔽端接元件，将所有屏蔽的电缆直接通过导轨连接接地。

2）必须在断电情况下对组件进行接线。

（3）警告

1）带电作业会有生命危险。

2）如果带电对组件的前插头进行接线，会有触电危险。

（4）其他信息

其他注意事项可参见手册“CPU 数据”手册以及 CPU 的安装手册。

四、参数赋值工具介绍

借助于“PID 参数设置”工具，可以很方便地调试功能块 SFB41/FB41、SFB42/FB42 的参数（背景数据块）。

1. 调试 PID 参数的用户界面

在 Windows 操作系统中，调用“调试 PID 参数用户界面”的操作过程如下：

Start→SIMATIC→STEP 7→PID Control Parameter Assignment，如图 6-74 所示。

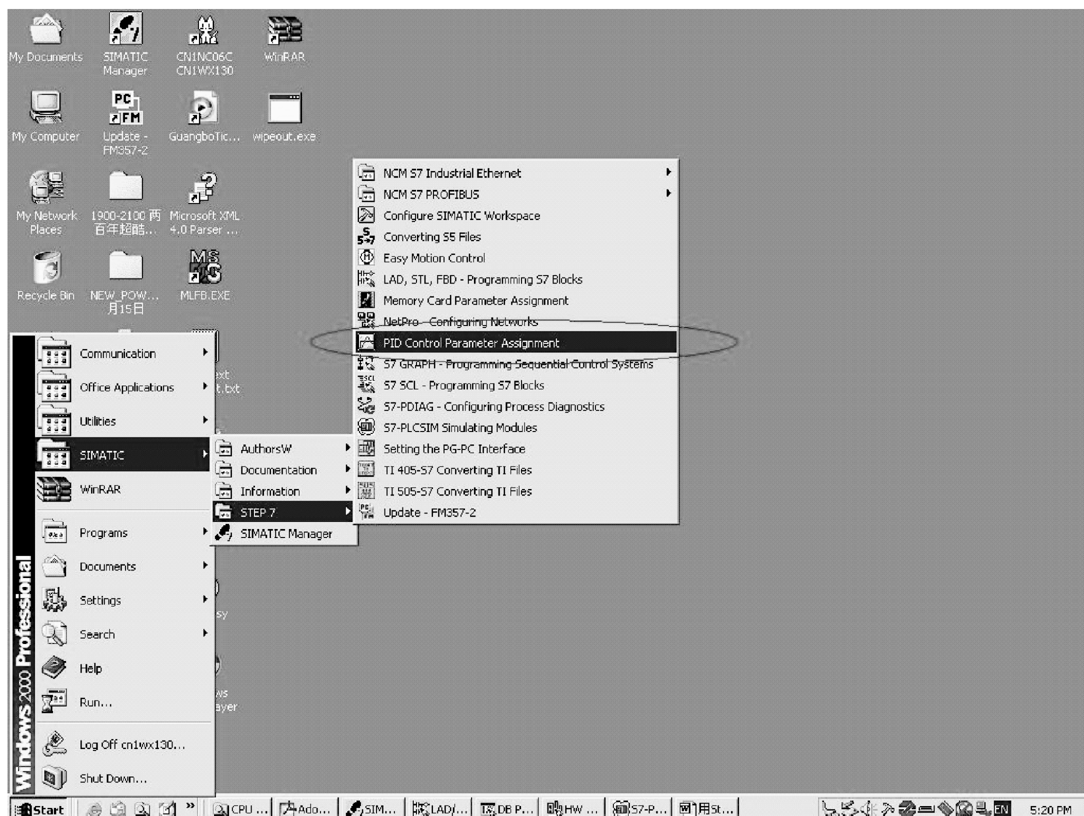


图 6-74　调用“调试 PID 参数用户界面”

1）在最开始的对话框中，既可以打开一个已经存在的 FB41/ SFB41“CONT_C”或者 FB42/ SFB42“CONT_S”的背景数据块，也可以生成一个新的数据块，还可以分配给 FB41/SFB41“CONT_C”或者 FB42/SFB42“CONT_S”，作为背景数据块。背景数据块的设置如图 6-75 所示。

2）FB43/SFB43“PULSEGEN”没有参数设置的用户界面工具。必须在 STEP 7 中去设置它的参数。

2. 获取在线帮助的途径

当分配参数给 FB41/SFB41“CONT_C”、FB42/SFB42“CONT_S”或者 FB43/SFB43“PULSEGEN”时，可以通过以下三条途径获得帮助：

1）使用 STEP7 菜单 Help→Contents，获得相应的帮助信息。

2）通过按下“F1”键得到帮助。

3）在 PID 参数设置对话框中，通过点击“Help”，可以得到具体的帮助信息。

五、在用户程序中实现

1. 调用功能块

使用相应的背景数据块调用系统功能块。

举例：CALL SFB 41，DB 30（或者，CALL FB 41，DB 31）。

2. 背景数据块

系统功能块的参数将保存在背景数据块中。在第 6 章中将介绍这些参数。

可以通过以下方式访问这些参数：

1）DB 编号和偏移地址。

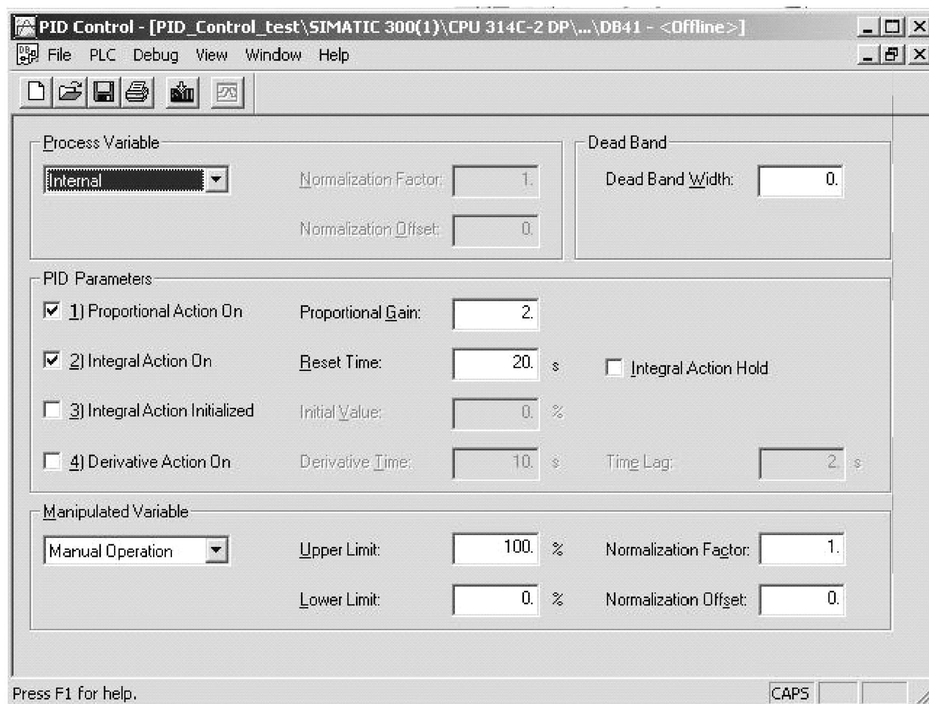


图 6-75 背景数据块的设置

2）数据块编号和数据块中的符号地址。

3. 程序结构

SFB 必须在重新起动组织块 OB100 中和循环中断组织块 OB30～38 中调用。

模式：

1）OB100 Call SFB/FB 41、42、43，DB 30。

2）OB35 Call SFB/FB 41、42、43，DB 30。

六、功能块介绍

1. 连续调节功能 SFB 41/FB 41 “CONT_C”

（1）简介

SFB/FB “CONT_C”（连续控制器）用于使用连续的 I/O 变量在 SIMATIC S7 控制系统中控制技术过程。可以通过参数打开或关闭 PID 控制器，以此来控制系统。通过参数赋值工具，可以很容易地做到这一点。调用：Start→SIMATIC→STEP 7→PID Control Parameter Assignment。在线电子手册，见 Start→SIMATIC→Documentation→English→STEP 7-PID Control（见图 6-76）。

（2）应用程序

可以使用控制器作为单独的 PID 定点控制器或在多循环控制中作为级联控制器、混合控制器和比例控制器使用。控制器的功能基于带有一个模拟信号的采样控制器的 PID 控制算法，如果必要的话，可以通过脉冲发送器（PULSEGEN）进行扩展，以产生脉冲宽度调制的输出信号，来控制比例执行机构的两个或 3 个步进控制器。

（3）说明

除了设定点操作和过程数值操作的功能以外，SFB 41/FB 41（CONT_C）可以使用连续的变量输出和手动影响控制数值选项，来实现一个完整的 PID 控制器。下面是关于 SFB 41/FB 41（CONT_C）的详细的子功能说明：

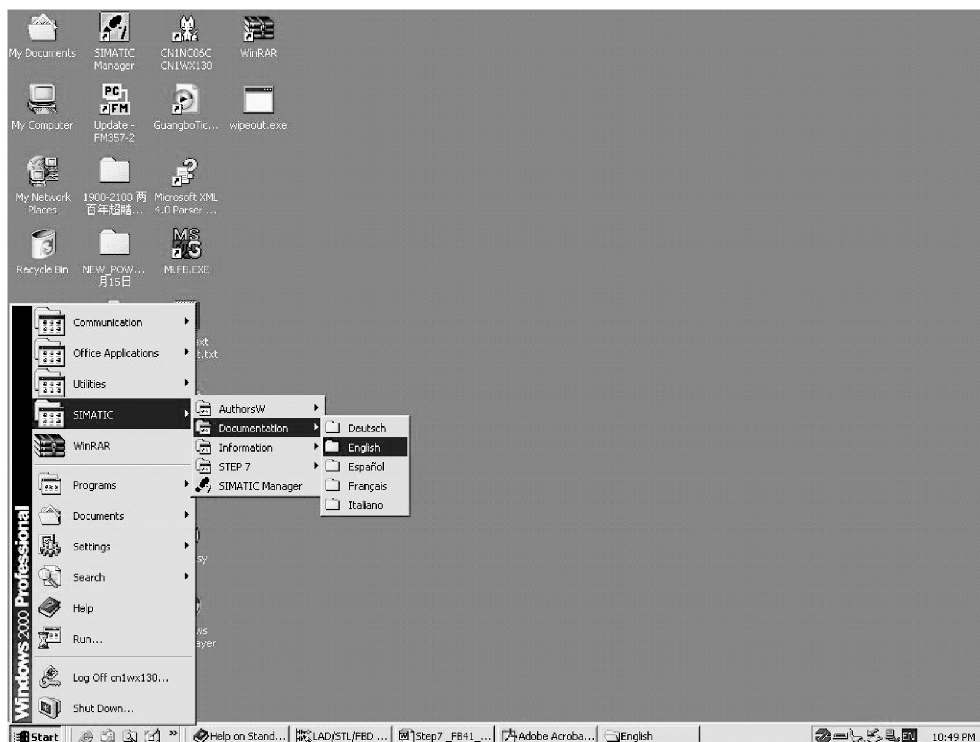


图 6-76　打开在线电子手册

1）设定点操作：设定点以浮点格式在“SP_INT”端输入。

2）实际数值操作：

过程变量可以在外围设备（I/O）或者以浮点数值格式输入。“CRP_IN”功能可以将“PV_PER”外围设备数值转换为一个浮点格式的数值，在 −100% ~ +100% 之间，转换公式如下：

CPR_IN 的输出 = PV_PER × 100/27648

“PV_NORM”功能可以根据下述规则标准化“CRP_IN”的输出：

输出 PV_NORM = (CPR_IN 的输出) × PV_FAC + PV_OFF

“PV_FAC”的默认值为“1”，“PV_OFF”的默认值为“0”。

变量“PV_FAC”和“PV_OFF”为下述公式转化的结果：

PV_OFF = (PV_NORM 的输出) − (CPR_IN 的输出) × PV_FAC

PV_FAC = （PV_NORM 的输出 − PV_OFF）/（CPR_IN 的输出）

不必转换为百分比数值。如果设定点为物理确定，实际数值还可以转换为该物理数值。

3）负偏差计算：设定点和实际数值之间的区别便形成负值偏差。为了抑制由于被控量的量化引起的小的、恒定的振荡（例如使用 PULSEGEN 进行脉冲宽度调制），在死区将施加一个死区。如果 DEADB_W = 0，则死区将关闭。

4）PID 算法：PID 算法作为一种位置算法进行控制。比例运算、积分运算（INT）和微商运算（DIF）都可并行连接，也可以单独激活或取消。这就允许组态成 P、PI、PD 和 PID 控制器。也可以是纯 I 和 D 调节器。

5）手动模式：可以在手动模式和自动模式之间切换。在手动模式下，被控量被修改成手动选定的数值。

积分器（INT）内部设置为“LMN-LMN_P-DISV”，微商器（DIF）内部设置为“0”，并进行内部匹配。这就是说切换到自动模式时不会引起被控量的突变。

6）受控数值的处理：使用 LMNLIMIT 功能，受控数值可以被限制为一个所选择的数值。当输入变量超出极限值时，信号位将指示。“LMN_NORM”功能可以根据下述公式标准化“LMN-LIMIT”的输出：

LMN =（LMNLIMIT 的输出）× LMN_FAC + LMN_OFF

“LMN_FAC”的默认值为“1”，“LMN_OFF”的默认值为“0”。

受控数值也适用于外围设备（I/O）格式。“CPR_OUT”功能可以将浮点值“LMN”转换为一个外围设备值，转换公式如下：

LMN_PER = LMN × 2764/10

7）前馈控制：一个干扰变量被引入“DISV”端输入。

8）初始化：SFB 41/FB 41“CONT_C”有一个初始化程序，可以在输入参数 COM_RST = TRUE 置位时运行。在初始化过程中，积分器可以内部设置为初始值“I_ITVAL”。如果在一个循环中断优先级调用它，它将从该数值继续开始运行。所有其他输出都设置为其默认值。

9）出错信息：故障输出参数 RET_VAL 不使用。

10）SFB/FB“CONT_C”（连续调节控制器）框图如图 6-77 所示。

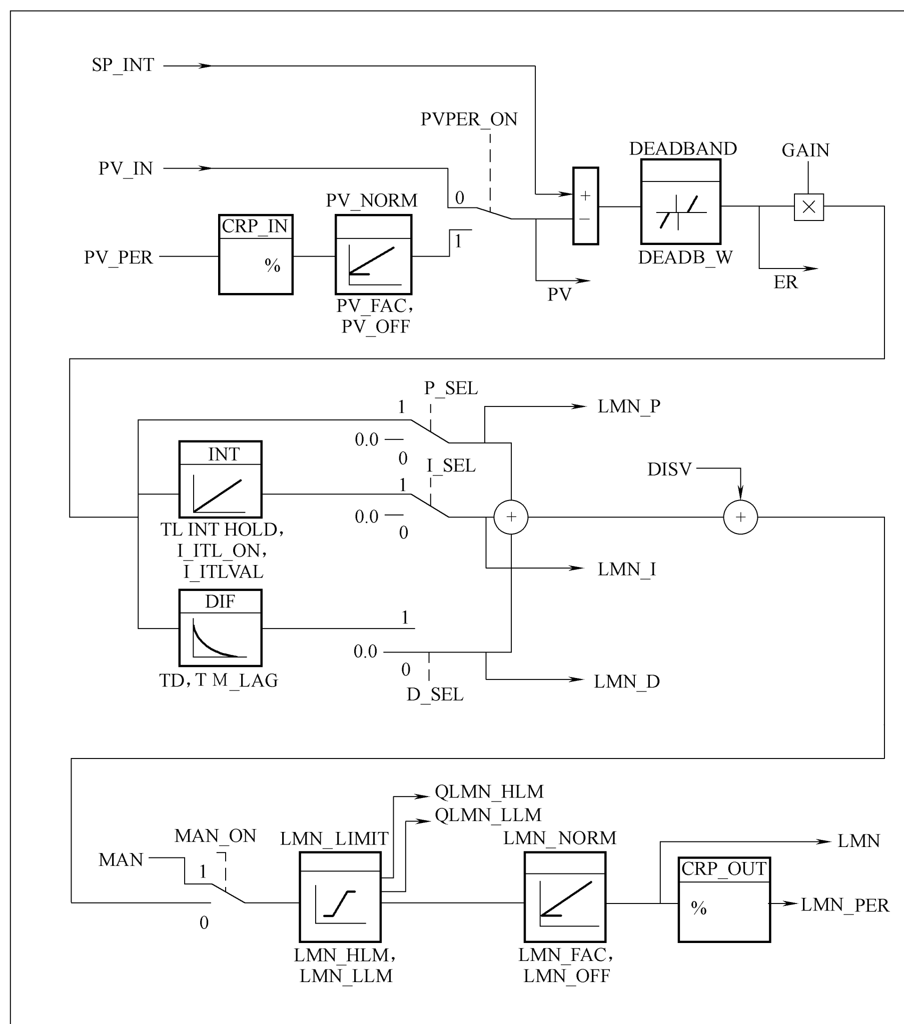


图 6-77 SFB/FB“CONT_C”框图

11）输入参数：SFB 41/FB 41 “CONT_C” 如图 6-78 所示。

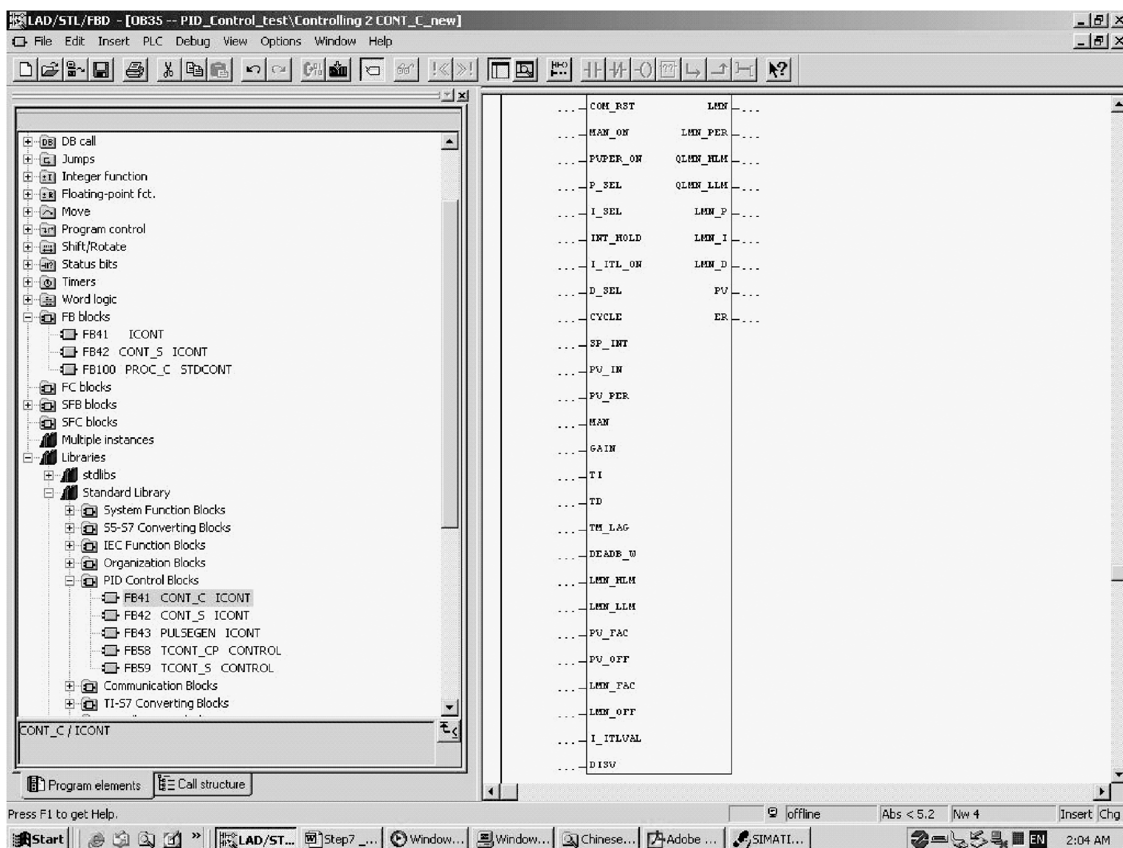


图 6-78　输入参数 SFB 41/FB 41 “CONT_C”

SFB 41/FB 41 “CONT_C” 输入参数的说明见表 6-15。

表 6-15　SFB 41/FB 41 “CONT_C” 输入参数的说明

序号	参数	数据类型	数值范围	默认	说　明
1	COM_RST	BOOL		FAULSE	COMPLETE RESTART(完全再起动) 该块有一个初始化程序,可以在输入参数 COM_RST 置位时运行
2	MAN_ON	BOOL		TRUE	MANUAL VALUE ON(手动数值接通) 如果输入端“手动数值接通”被置位,那么闭环控制循环将中断。手动数值被设置为受控数值
3	PVPER_ON	BOOL		FALSE	PROCESS VARIABLE PERIPHERY ON/(过程变量外设接通) 如果过程变量从 I/O 读取,输入“PV_PER”必须连接到外围设备,并且输入“ PROCESS VARIABLE PERIPHERY ON”必须置位
4	P_SEL	BOOL		TRUE	PROPORTIONAL ACTION ON(比例分量接通) PID 各分量在 PID 算法中可以分别激活或者取消。当输入端“比例分量接通”被置位时,P 分量被接通
5	I_SEL	BOOL		TRUE	INTEGRAL ACTION ON(积分分量接通) PID 各分量在 PID 算法中可以分别激活或者取消。当输入端“积分分量接通”被置位时,I 分量被接通

（续）

序号	参数	数据类型	数值范围	默认	说　明
6	INT_HOLD	BOOL		FALSE	INTEGRAL ACTION HOLD（积分分量保持） 积分器的输出被冻结。为此，必须置位输入“Integral Action Hold（积分操作保持）”
7	I_ITL_ON	BOOL		FALSE	INITIALIZATION OF THE INTEGRAL ACTION（积分分量初始化接通） 积分器的输出可以被设置为输入“I_ITLVAL”。为此，必须置位输入“积分操作的初始化”
8	D_SEL	BOOL		FALSE	DERIVATIVE ACTION ON（微分分量接通） PID 各分量在 PID 算法中可以分别激活或者取消。当输入端“微分分量接通”被置位时，D 分量被接通
9	CYCLE	TIME	≥1ms	T#1s	SAMPLE TIME（采样时间） 块调用之间的时间必须恒定。“采样时间”输入规定了块调用之间的时间，应该与 OB35 设定时间保持一致
10	SP_INT	REAL	－100.0 ~ ＋100.0（%） 或者物理值 1	0.0	INTERNALSETPOINT（内部设定点） 内部设定点”输入端用于确定设定值
11	PV_IN	REAL	－100.0 ~ ＋100.0（%） 或者物理值 1	0.0	PROCESSVARIABLE IN（过程变量输入） 可以设置一个初始值到“过程变量输入”端或者连接一个浮点数格式的外部过程变量
12	PV_PER	WORD		W#16#0000	PROCESS VARIABLE PERIPHERY（过程变量外设） 外围设备的实际数值，通过 I/O 格式的过程变量被连接到“过程变量外围设备”输入端，连接到控制器
13	MAN	REAL	－100.0 ~ ＋100.0（%） 或者物理值 2	0.0	MANUAL VALUE（手动数值） “手动数值”输入端可以用于通过操作者接口功能设置一个手动数值
14	GAIN	REAL		2.0	PROPORTIONAL GAIN（比例增益） “比例增益”输入端可以设置控制器的比例增益系数
15	TI	TIME	≥CYCLE	T#20s	RESET TIME（复位时间） “复位时间”输入端确定了积分器的时间响应
16	TD	TIME	≥CYCLE	T#10s	DERIVATIVE TIME（微分时间） “微分时间”输入端确定了微分单元的时间响应
17	TM_LAG	TIME	≥（CYCLE/2）	T#2s	TIME LAG OF THE DERIVATIVE ACTION（微分分量的滞后时间） 微分操作的算法包括一个时间滞后，可以被赋值给“微分分量的滞后时间”输入端
18	DEADB_W	REAL	≥0.0（%） 或者物理值 1	0.0	DEAD BAND WIDTH（死区宽度） 死区用于存储错误。“死区宽度”输入端确定了死区的容量大小
19	LMN_HLM	REAL	LMN_LLM ~ 100.0（%） 或者物理值 2	100.0	MANIPULATED ALUE HIGH LIMIT（受控数值的上限） 受控数值必须设定有一个“上限”和一个“下限”。“受控数值上限”输入端确定了“上极限”
20	LMN_LLM	REAL	－100.0（%）~ LMN_HLM 或者物理值 2	0.0	MANIPULATED VALUE LOW LIMIT（受控数值的下限） 受控数值必须设定有一个“上限”和一个“下限”。“受控数值下限”输入端确定了“下极限”

（续）

序号	参数	数据类型	数值范围	默认	说　明
21	PV_FAC	REAL		1.0	PROCESS VARIABLE FACTOR(过程变量系数) “过程变量系数”输入端用于和过程变量相乘。该输入端可以用于匹配过程变量范围
22	PV_OFF	REAL		0.0	PROCESSVARIABLE OFFSET(过程变量偏移量) “过程变量偏移”输入端可以添加到“过程变量”。该输入端可以用于匹配过程变量的范围
23	LMN_FAC	REAL		1.0	MANIPULATED VALUE FACTOR(受控数值系数) “受控数值系数”输入端用于与受控数值相乘。该输入端可以用于匹配受控数值的范围
24	LMN_OFF	REAL		0.0	MANIPULATED VALUE(受控数值的偏移量) “受控数值的偏移量”可以与受控数值相加。该输入端可以用于匹配受控数值的范围
25	I_ITLVAL	REAL	-100.0 ~ +100.0(%) 或者物理值2	0.0	INITIALIZATION VALUE OF THE INTEGRAL-ACTION(积分分量初始化值) 积分器的输出可以用输入端“I_ITL_ON”设置。初始化数值可以设为“积分分量初始值”输入
26	DISV	REAL	-100.0 ~ +100.0(%) 或者物理值2	0.0	DISTURBANCE VARIABLE(干扰变量) 对于前馈控制,干扰变量被连接到“干扰变量”输入端

注：1. 设定值通道”和“过程变量通道”中的参数，应该有相同的单位。例如，如果使用 PV_IN 作为“过程物理值”或者“过程物理值百分比”，SP_INT 必须使用相同的单位；如果使用 PV_PER 作为外围设备的实际数值，SP_INT 只能使用“-100.0 ~ +100.0（%）”作为设定值。如果设定值是 SP_INT 是 0 ~ 10MPa 中的 8MPa，那么需要填写 0.8，PV_PER 填写硬件外设地址 IW XXX。
2. 受控量通道中的参数应该有相同的单位。

12）输出参数：SFB 41/FB 41“CONT_C”输出参数的说明见表 6-16。

表 6-16　SFB 41/FB 41“CONT_C”输出参数的说明

序号	参数	数据类型	数值范围	默认	说　明
1	LMN	REAL		0.0	MANIPULATED VALUE(受控数值) 有效的受控数值被以浮点数格式输出在“受控数值”输出端上
2	LMN_PER	WORD		W#16#0000	MANIPULATEDVALUE PERIPHERY(受控数值外围设备) I/O 格式的受控数值被连接到“受控数值外围设备”输出端上的控制器
3	QLMN_HLM	BOOL		FALSE	HIGH LIMIT OF MANIPULATED VALUE REACHED(达到受控数值上限)受控数值必须规定一个最大极限和一个最小极限。“达到受控数值上限”指示已超过最大极限
4	QLMN_LLM	BOOL		FALSE	LOW LIMIT OF MANIPULATED VALUE REACHED(达到受控数值下限) 受控数值必须规定一个最大极限和一个最小极限。“达到受控数值下限”指示已超过最小极限
5	LMN_P	REAL		0.0	PROPORTIONALITY COMPONENT(比例分量) “比例分量”输出端输出受控数值的比例分量
6	LMN_I	REAL		0.0	INTEGRAL COMPONENT(积分分量) “积分分量”输出端输出受控数值的积分分量
7	LMN_D	REAL		0.0	DERIVATIVE COMPONENT(微分分量) “微分分量”输出端输出受控数值的微分分量

（续）

序号	参数	数据类型	数值范围	默认	说　　明
8	PV	REAL		0.0	PROCESS VARIABLE(过程变量) 有效的过程变量在“过程变量”输出端上输出
9	ER	REAL		0.0	ERROR SIGNAL(误差信号) 有效误差在“误差信号”输出端输出

2. 步进控制功能 SFB 42/FB 42 “CONT_S”

（1）简介

SFB/FB “CONT_S”（步进控制器）用在 SIMATIC S7 PLC 上，用于二进制数控数值输出信号积分执行机构的控制技术过程。在参数赋值过程中，可以激活或取消 PI 步进控制器的子功能，以使控制器与过程匹配。通过参数赋值工具，可以很容易地做到这一点。调用：Start→SIMATIC→STEP 7→PID Control Parameter Assignment。在线电子手册，见 Start→SIMATIC→Documentation→English→STEP 7-PID Control。

（2）应用程序

可以使用该控制器作为单独的 PI 固定设定值控制器，或者在辅助控制循环（第二级闭环）中作为级联控制器、混合控制器或者比例控制器使用，但是不能用作主控制器（第一级调节器）。控制器的功能根据采样控制器的 PI 控制算法实现，由模拟执行信号生成二进制输出信号。

下列功能适用于 CPU 314 IFM 的 FB V1.5 或 V1.1.0 以上版本：

利用 TI = T#0ms，可以封锁调节器的积分分量。因此，允许功能块用作比例（P）控制器。

由于控制器不使用任何位置反馈信号，内部计算的受控变量将不能准确地匹配信号控制元件的位置。如果受控变量（ER*GAIN）为负值，应进行调整。然后调节器置位输出端 QLMNDN（受控量信号低），直到 LMNR_LS（位置反馈信号下限）被置位。

控制器还可以在一个控制器级联中用作一个辅助控制器（第二个执行器）。设定点输入端“SP_INT”用于赋值控制元件的位置。在这种情况下，实际数值输入和参数“TI（积分时间）”必须被设置为“0”。一个应用实例如下：通过电控阀瓣控制温度，即借助二进制脉冲数值输出信号来控制热量输出的温度调节和利用阀门控制制冷容量。在这种情况下，为了关闭全部阀门，受控变量（ER*GAIN）应该有一个负值。

（3）说明

除了过程数据通道的功能外，SFB/FB “CONT_S”（步进控制器）可以使用一个数字受控数值输出和手动影响控制数值选项，来实现一个完整的 PI 控制器。步进控制器不使用位置反馈信号。限位信号可以用于限制脉冲输出。下面是详细的子功能说明：

1）设定点操作：设定点以浮点数格式在“SP_INT”输入端上输入。

2）实际数值操作：过程变量可以在外围设备（I/O）或以浮点数格式输入。“CRP_IN”功能可以将“PV_PER”外围设备数值转换为一个浮点数格式的数值，在 -100% ~ +100% 之间，转换公式如下：

CPR_IN 的输出 = PV_PER × 100/27648

“PV_NORM”功能可以根据下述公式标准化“CRP_IN”的输出：

PV_NORM 的输出 =（CPR_IN 的输出）× PV_FAC + PV_OFF

PV_FAC 的默认值为“1”，PV_OFF 的默认值为“0”。

变量“PV_FAC”和“PV_OFF”为下述公式转化的结果：

PV_OFF =（PV_NORM 的输出）-（CPR_IN 的输出）× PV_FAC

PV_FAC =（PV_NORM 的输出 – PV_OFF）/（CPR_IN 的输出）

3）负偏差计算：设定点和实际数值之间的区别便形成负值偏差。为了抑制由于受控变量的量化 造成的小的、恒定的振荡（例如，由于执行机构阀门引起的受控数值的波动），为负偏差设置了一个死区。如果 DEADB_W = 0，则死区将被关闭。

4）PI 步进算法：SFB/FB“CONT_S”（步进控制器）不使用位置反馈信号。PI 算法的积分操作和假定位置反馈信号都在积分器（INT）中计算，并作为一个反馈值与剩余 P 操作进行比较。比较差被用于一个三步元件（THREE_ST）和一个脉冲发生器（PULSEOUT），以生成执行机构的控制脉冲。控制器的开关频率可以通过在三步元件上采用阈值控制来减少。

5）前馈控制：一个干扰变量被引入“DISV”输入端。

6）初始化操作：SFB/FB“CONT_S”（步进控制器）有一个初始化程序，可以在输入参数 COM_RST = TRUE 置位时运行。所有其他输出端都设置为其默认值。

7）出错信息：故障输出参数 RET_VAL 不使用。

8）SFB/FB“CONT_S”（步进控制器）框图如图 6-79 所示。

9）输入参数：SFB 42/FB 42“CONT_S”如图 6-80 所示。

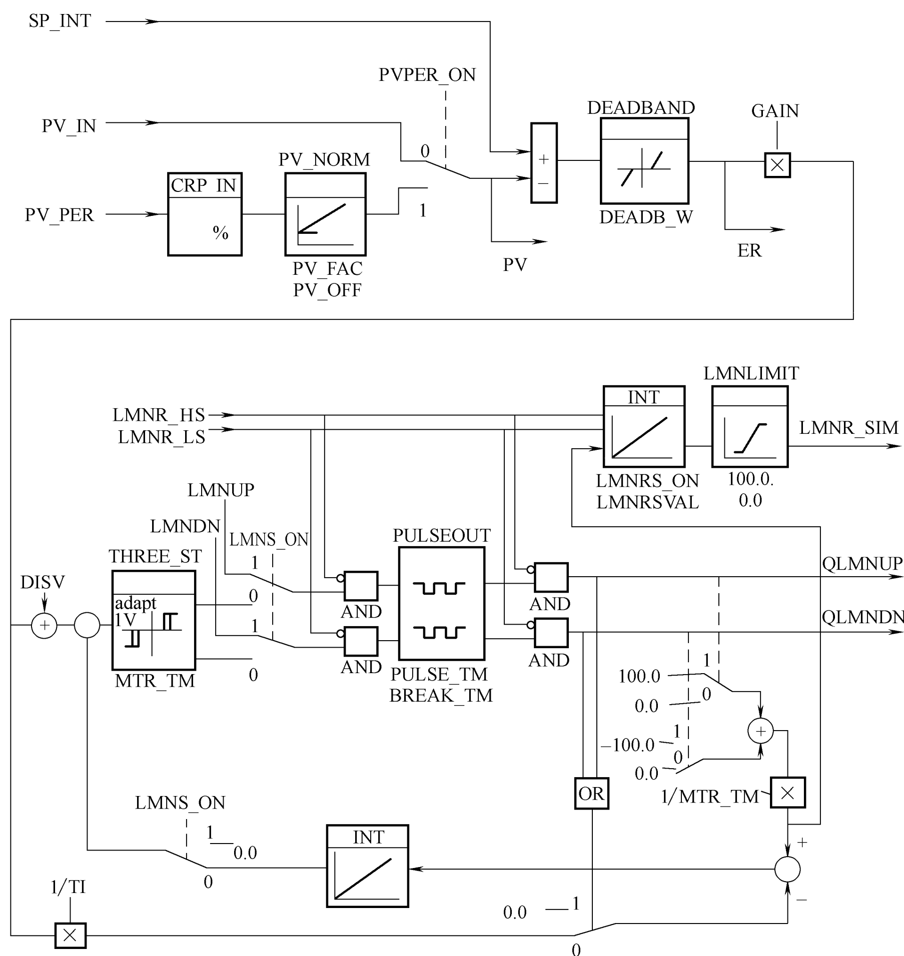


图 6-79　SFB/FB“CONT_S”框图

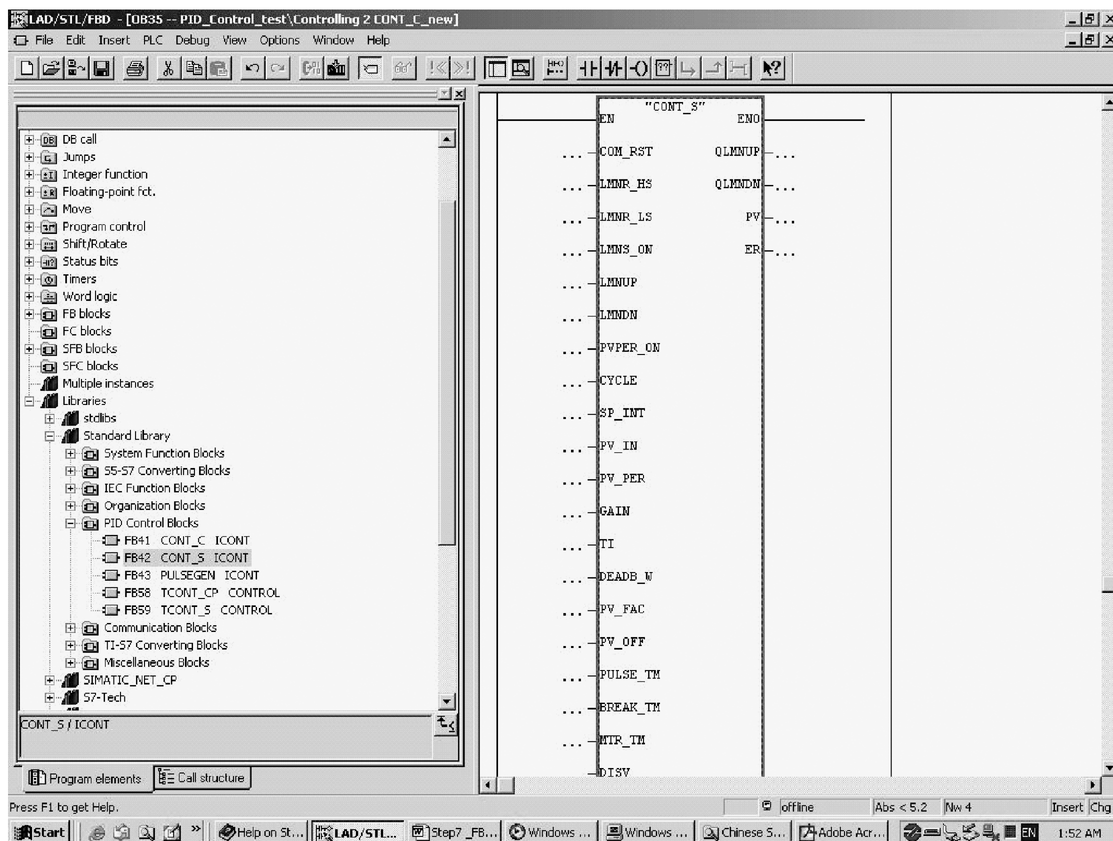


图 6-80 输入参数 SFB 42/FB 42 “CONT_S”

SFB 42/FB 42 “CONT_S” 输入参数的说明见表 6-17。

表 6-17 SFB 42/FB 42 “CONT_S” 输入参数的说明

序号	参数	数据类型	数值范围	默认	说 明
1	COM_RST	BOOL		FAULSE	COMPLETE RESTART(完全再起动) 该块有一个初始化程序,可以在输入参数 COM_RST 置位时运行
2	LMNR_HS	BOOL		FALSE	HIGH LIMIT OF POSITION FEEDBACK SIGNAL(位置反馈信号上限) “执行器在上限停”信号连接到“位置反馈信号上限”输入端。LMNR_HS = TRUE 表示执行器处于最大上限
3	LMNR_LS	BOOL		FALSE	LOW LIMIT OF POSITION FEEDBACK SIGNAL(位置反馈信号下限) “执行器在下限停”信号连接到“位置反馈信号下限”输入端。LMNR_LS = TRUE 表示执行器处于最大下限
4	LMNS_ON	BOOL		TRUE	MANUAL ACTUATING SIGNALS ON(手动执行信号接通)通过“手动执行信号接通”执行信号处理切换为手动模式
5	LMNUP	BOOL		FALSE	ACTUATING SIGNALS UP(执行信号上升) 通过手动执行信号,输出信号“QLMNUP”在“执行信号上升沿”输入被置位

（续）

序号	参数	数据类型	数值范围	默认	说明
6	LMNDN	BOOL		FALSE	ACTUATING SIGNALS DOWN(执行信号下降) 通过手动执行信号,输出信号“QLMNDN”在“执行信号下降沿”输入被置位
7	PVPER_ON	BOOL		FALSE	PROCESS VARIABLE PERIPHERY ON(过程变量外设接通) 如果从I/O读取过程变量,输入端“PV_PER”必须连接到外围设备,并且输入端“PROCESS VARIABLE PERIPHERY ON”必须置位
8	CYCLE	TIME	≥1ms	T#1s	SAMPLING TIME(采样时间) 块调用之间的时间必须恒定。“采样时间”输入端规定了块调用之间的时间
9	SP_INT	REAL	-100.0 ~ +100.0(%) 或物理值1	0.0	INTERNAL SETPOINT(内部设定值) “内部设定值”输入用于确定一个设定值
10	PV_IN	REAL	-100.0 ~ +100.0(%) 或物理值1	0.0	PROCESS VARIABLE IN(过程变量输入) 可以设置一个初始值到“过程变量输入”端或者连接一个浮点数格式的外部过程变量
11	PV_PER	WORD		W#16#0000	PROCESS VARIABLE PERIPHERY(过程变量外设) I/O格式的过程变量被连接到调节器的“过程变量外围设备”输入端
12	GAIN	REAL		2.0	PROPORTIONAL GAIN(比例增益) “比例增益”输入端设置控制器的增益
13	TI	TIME	≥CYCLE	T#20s	RESET TIME(复位时间) “复位时间”输入端确定了积分器的时间响应
14	DEADB_W	REAL	0.0 ~ +100.0(%) 或物理值1	1.0	DEAD BAND WIDTH(死区宽度) 死区用于误差。“死区宽度”用于确定死区的大小
15	PV_FAC	REAL		1.0	PROCESS VARIABLE FACTOR(过程变量系数) “过程变量系数”输入用于和过程变量相乘。该输入可以用于匹配过程变量的范围
16	PV_OFF	REAL		0.0	PROCESS VARIABLE OFFSET(过程变量偏移量) “过程变量偏移”输入端与过程变量相加。该输入端用于匹配过程变量的范围
17	PULSE_TM	TIME	≥CYCLE	T#3 s	MINIMUM PULSE TIME(最小脉冲时间) 最小脉冲宽度可以使用参数“最小脉冲时间”赋值
18	BREAK_TM	TIME	≥CYCLE	T#3 s	MINIMUM BREAK TIME(最小间隔时间) 最小脉冲间隔时间可以使用参数“最小间隔时间”赋值
19	MTR_TM	TIME	≥CYCLE	T#30 s	MOTOR MANIPULATED VALUE(电动执行时间) 执行机构从一个限幅位置移动到另一个限幅位置所需的时间,可以在参数“电动执行时间”参数中输入
20	DISV	REAL	-100.0 ~ +100.0(%) 或物理值2	0.0	DISTURBANCE VARIABLE(干扰变量) 对于前馈控制,干扰变量连接到输入端“干扰变量”

注：1. “设定值通道” 和 “过程变量通道” 中的参数，应该有相同的单位。
2. 受控量通道中的参数应该有相同的单位。

10）输出参数：SFB 42/FB 42 “CONT_S” 输出参数的说明见表 6-18。

表 6-18　SFB 42/FB 42 “CONT_S” 输出参数的说明

序号	参数	数据类型	数值范围	默认	说　明
1	QLMNUP	BOOL		FALSE	ACTUATING SIGNAL UP(执行信号上升) 如果输出端“执行信号上升”被置位,那么执行阀是打开的
2	QLMNDN	BOOL		FALSE	ACTUATING SIGNAL DOWN(执行信号下降) 如果输出端“执行信号下降”被置位,那么执行阀是打开的
3	PV	REAL		0.0	PROCESS VARIABLE(过程变量) 有效的过程变量是在“过程变量”输出端输出。
4	ER	REAL		0.0	ERROR SIGNAL(负偏差信号) 有效的负偏差数值在“负偏差信号”输出端输出

3. 脉冲宽度调制器 SFB 43/FB 43 “PULSEGEN”

（1）简介

SFB/FB “PULSEGEN”（脉冲发生器）可以用于为 PID 控制器使用比例执行机构的脉冲输出。在线电子手册，见 Start→SIMATIC→Documentation→English→STEP 7-PID Control。

（2）应用程序

使用 SFB/FB “PULSEGEN”（脉冲发生器），可以通过脉冲宽度调制，组态 PID 两步或三级控制器。该功能一般与连续控制器 SFB/FB “CONT_C” 一起使用（见图 6-81）。

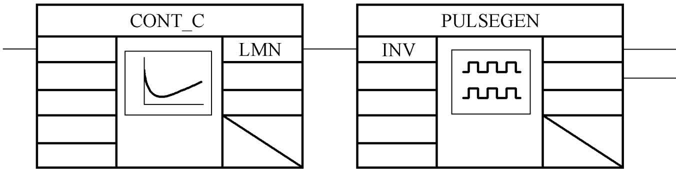


图 6-81　PULSEGEN 与 CONT_C 一起使用

（3）说明

“PULSEGEN” 可以通过调制脉冲宽度，将输入变量 “INV”（= PID 控制器的 LMN）转换为一个恒定周期的脉冲串，该恒定周期相当于输入变量刷新的循环时间，必须在 “PER_TM” 中赋值。

每个周期的脉冲宽度与输入变量成正比。“PER_TM” 中的循环时间与 SFB/FB “PULSEGEN” 的处理时间不同。“PER_TM” 循环时间由多个 SFB/FB “PULSEGEN” 执行循环之和。因此，每个 “PER_TM” 循环的 SFB/FB “PULSEGEN” 调用次数是脉冲宽度，可以精确测量脉冲宽度。最小受控数值在参数 “P_B_TM” 中确定（见图 6-82）。

1）脉冲宽度调制：

输入变量 30% 以及每个 PER_TM 循环时间调用 SFB/FB “PULSEGEN” 10 次，含义如下：

- 对于前 3 个 SFB/FB “PULSEGEN”（10 次调用的 30 %），输出 “QPOS” 为 “1”；
- 对于其余 7 个 SFB/FB “PULSEGEN”（10 次调用的 70 %），输出 “QPOS” 为 “0”。

2）SFB/FB “PULSEGEN” 块图如图 6-83 所示。

3）受控数值的精度：如果 “采样频率比例” 为 1∶10（“CONT_C” 调用与 “PULSEGEN” 调用之比），那么在这个例子中受控数值的精度降低为 10 %。换句话说，设定的输入数值 “INV”

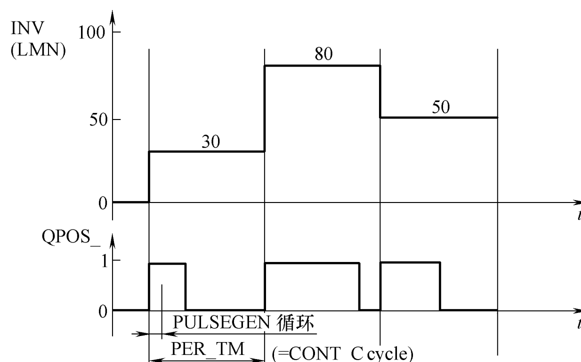


图 6-82 PULSEGEN 的使用

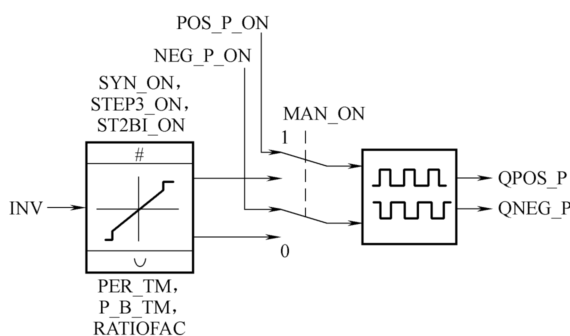


图 6-83 SFB/FB“PULSEGEN”块图

只能在“QPOS”输出端上以“10%”的步长转换成脉冲宽度。

只有当每次“CONT_C”调用中“PULSEGEN”调用的次数增加时，才能提高精度。

例如，如果每个“CONT_C”调用的“PULSEGEN”调用次数为 100，受控数值的分辨率将达到 1%（建议分辨率≤5%）。

注意：“采样频率比例”必须由用户编程。

4）自动同步：可以使刷新输入变量“INV”的块（例如，“CONT_C”），与脉冲输出自动同步。这就保证了输入变量中的一个变化可以尽可能快地输出为一个脉冲。

脉冲发生器可以根据“PER_TM”的周期为时间间隔，定期评价输入数值“INV”，并将该数值转换为相应长度的脉冲信号。

但是，由于“INV”一般在较慢的循环中断级中计算，所以脉冲发生器应在“INV”刷新后尽可能快地将具体数值转换为一个脉冲信号。

为此，块必须使用下述程序对周期的起点同步：

如果“INV”变化，并且块调用不在一个周期的第 1 个或最后两个调用循环中，可以进行同步。将重新计算脉冲宽度，并在下一个循环中输出一个新的周期（见图 6-84）。

自动同步可以根据“SYN_ON”（=FALSE）输入关闭。

注意：在一个周期的开始，“INV”（即 LMN）的先前数值的映像将被或多或少的混合到脉冲信号中。

5）PID 控制器输出工作模式：根据脉冲发生器所赋值的参数，可以将 PID 调节器组态成具有一个三级输出或者一个两向或单向的两极输出 PID 控制器。可能模式的开关组合设置见表 6-19。

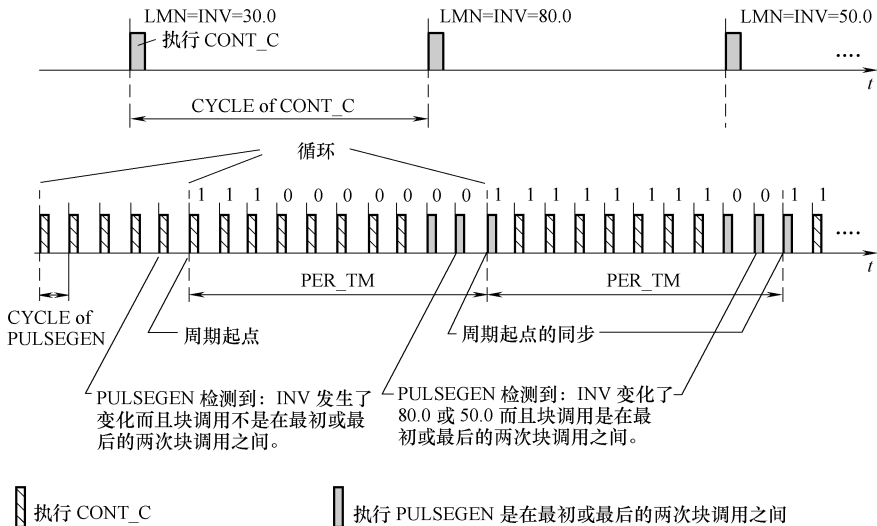


图 6-84 同步周期的起点

表 6-19 可能模式的开关组合设置

模 式	MAN_ON	STEP3_ON	ST2BI_ON
三级调节	FALSE	TRUE	ANY
两级调节，带双向调节区（－100% ～ ＋100%）	FALSE	FALSE	TRUE
两级调节，带单向调节区（0 ～ ＋100%）	FALSE	FALSE	FALSE
手动模式	TRUE	ANY	ANY

（4）三级控制

在“三级控制”模式下，可以生成控制信号的三种状态。二进制输出信号“QPOS_ P”和“QNEG_ P”的数值可以赋值给执行机构的状态。

温度控制的例子见表 6-20。

表 6-20 温度控制的例子

输出信号	加热	执行器关闭	制冷
QPOS_P	TRUE	FLASE	FLASE
QNEG_P	FLASE	FLASE	TRUE

根据输入变量，使用一个特性曲线可以计算脉冲宽度。特性曲线的形状取决于最小脉冲时间或最大中断时间和比例系数。

比例系数的正常值为“1”。

曲线中的“拐点”是由于最小脉冲时间或最小中断时间造成的。

最小脉冲或最小间隔时间

正确赋值最小脉冲或最小中断时间“P_B_TM”，可以防止短促的开断时间，以防降低开关元件和执行机构的使用寿命。

注意：否则，会删除一个短于“P_B_TM”脉冲宽度的输入变量“LMN”的较小绝对值。可以生成脉冲宽度大于“PER_TM-P_B_TM”的较大输入值被设置为 100% 或-100 %。

正脉冲宽度和负脉冲宽度可以根据输入变量（单位［%］）和周期时间相乘进行计算。

脉冲周期＝INV/100×PER_TM

图 6-85 所示为一个三级控制器的系统曲线（比例系数＝1）。

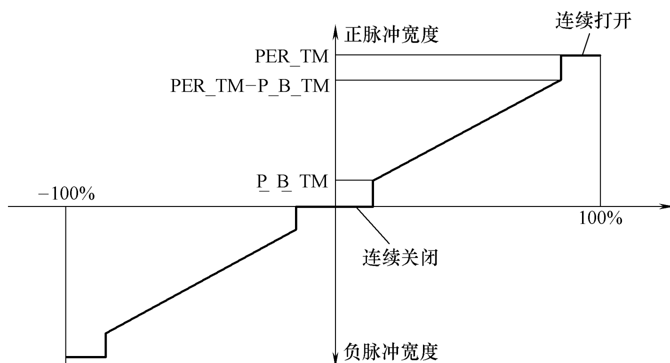


图 6-85　三级控制器的系统曲线（比例系数 =1）

使用比例系数“RATIOFAC”，可以改变正脉冲宽度和负脉冲宽度之比。例如，对于热处理，可用不同的时间常数加热和冷却执行机构。

比例系数也会影响最小脉冲/暂停周期。比例系数 <1 是指负脉冲的阈值乘以比例系数。

比例系数 <1

通过输入数值乘以脉冲周期所计算的比例系数，可以减少负脉冲输出的脉冲周期。

正脉冲周期 = INV/100 × PER_TM

负脉冲周期 = INV/100 × PER_TM x RATIOFAC

图 6-86 所示为一个三级控制器的系统曲线（比例系数 =0.5）。

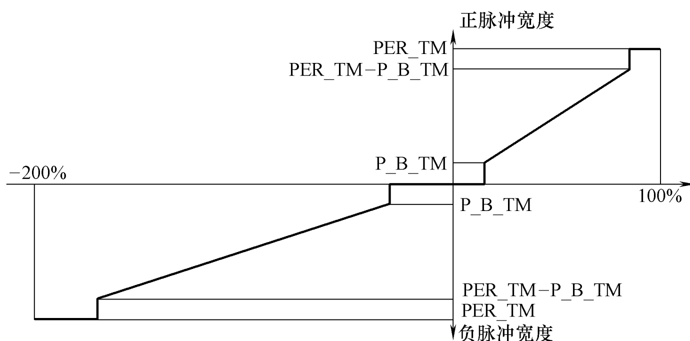


图 6-86　三级控制器的系统曲线（比例系数 =0.5）

比例系数 >1

通过输入数值乘以脉冲周期所计算的比例系数，可以减少正脉冲输出的脉冲周期。

负脉冲周期 = INV/100 × PER_TM

正脉冲周期 = INV/100 × PER_T/RATIOFAC

（5）二级控制

对于二级控制，只能将 PULSEGEN 的正脉冲输出“QPOS_P”连接到 I/O 执行机构。根据所使用的受控数值范围，二级控制器可以有一个双极或单极受控数值范围。

两级调节，带双向调节区（−100% ~ +100%）如图 6-87 所示。

两级调节，带单向调节区（0 ~ +100%）如图 6-88 所示。

如果控制循环中二级控制器的连接需要一个执行脉冲的逻辑转换二进制信号，可以在“QNEG_P”将输出信号进行“非”运算。

二级控制或三级控制中的手动模式

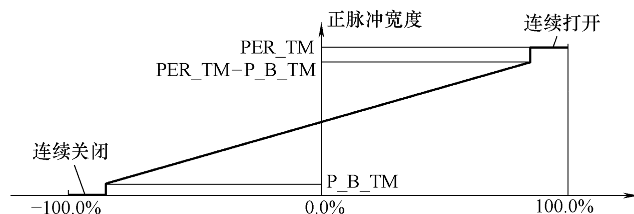


图 6-87 两级调节，带双向调节区（－100% ~ ＋100%）

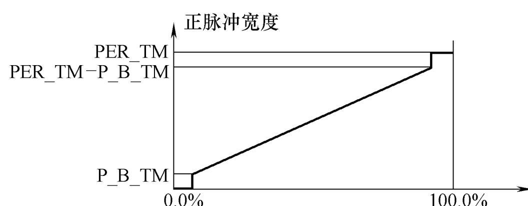


图 6-88 两级调节，带单向调节区（0 ~ ＋100%）

在手动模式（MAN_ON＝TRUE）中，三级控制器或二级控制器的二进制输出可以使用信号“POS_P_ON”和“NEG_P_ON”以及“INV”进行设置。

1）初始化：SFB“PULSGEN”有一个初始化程序，可以在输入参数 COM_RST＝TRUE 置位时运行。所有信号都被设置为“0”。

2）出错信息：故障输出参数 RET_VAL 不使用。

3）输入参数：SFB 43/FB 43“PULSEGEN”如图 6-89 所示。

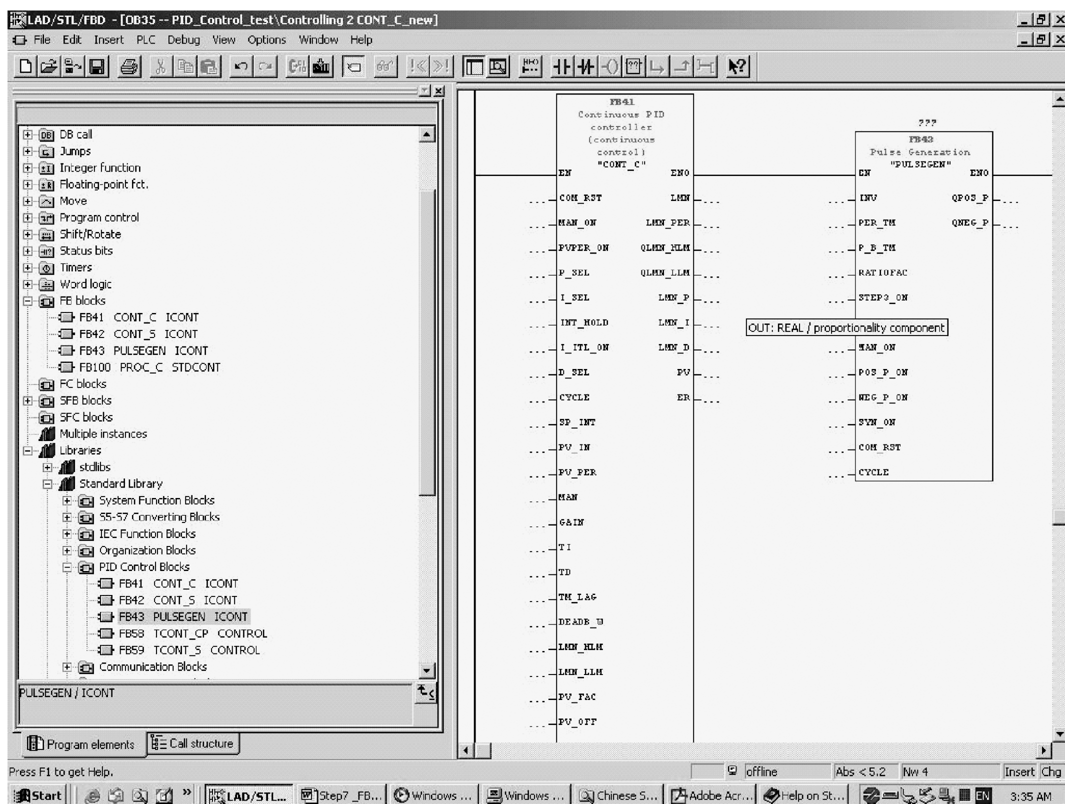


图 6-89 输入参数：SFB 43/FB 43“PULSEGEN”

SFB 43/FB 43 “PULSEGEN” 输入参数的说明见表 6-21。

表 6-21　SFB 43/FB 43 “PULSEGEN” 输入参数的说明

序号	参数	数据类型	数值范围	默认	说　明
1	INV	REAL	-100.0 ~ 100.0（%）	0.0	INPUT VARIABLE（输入变量） 模拟受控量连接到输入参数“输入变量” ● 对于 RATIOFAC < 1 的三级控制 ● 对于 RATIOFAC > 1 的三级控制 ● 对于双极二级控制 ● 对于多极二级控制
2	PER_TM	TIME	≥20 * CYCLE	T#1s	PERIOD TIME（周期时间） 脉冲宽度调制的恒定周期可以使用该输入参数输入。这相当于“CONT_C”控制器的采样时间。脉冲发生器的采样时间和“CONT_C”控制器的采样时间之比决定了脉冲宽度调制的精度
3	P_B_TM	TIME	≥CYCLE	T#0ms	MINIMUM PULSE/BREAK TIME（最小脉冲/间隔时间） 最小脉冲时间或最小中断时间可以使用输入参数“最小脉冲/间隔时间”赋值
4	RATIOFAC	REAL	0.1 ~ 10.0	1.0	RATIO FACTOR（比例系数） 输入参数“比例系数”可以用于改变正脉冲宽度和负脉冲宽度之比。例如，在热处理中，这可用于补偿加热和冷却的不同时间常数（例如，电加热和水冷过程）
5	STEP3_ON	BOOL		TRUE	THREE STEP CONTROL ON（三级调节接通） 该输入参数激活“三级调解”。在三级调节中，两路输出信号都被激活
6	ST2BI_ON	BOOL		FALSE	TWO STEP CONTROL FOR BIPOLAR MANIPULATED VALUE RANGE ON（两极调节，双向受控量范围接通） 用于双极受控数值范围打开的二级控制。可以在“双极受控数值”和“多极受控数值范围的二级控制”模式之间选择。此时，STEP3_ON = FALSE
7	MAN_ON	BOOL		FALSE	MANUAL MODE ON（手动模式接通） 通过设置该输入参数，可以手动设置输出信号
8	POS_P_ON	BOOL		FALSE	POSITIVE PULSE ON（正脉冲接通） 在三级控制的手动模式中，输出信号“QPOS_P”可以使用该输入参数进行控制。在二级控制的手动模式中，“QNEG_P”必须设置为“QPOS_P”相反
9	NEG_P_ON	BOOL		FALSE	NEGATIVE PULSE ON（负脉冲接通） 在三级控制的手动模式中，输出信号“QNEG_P”可以使用该输入参数进行控制。在二级控制的手动模式中，“QNEG_P”必须设置为“QPOS_P”相反
10	SYN_ON	BOOL		TRUE	SYNCHRONIZA TION ON（同步接通） 通过设置该输入参数，可以自动与刷新输入变量“INV”的块进行同步操作。这可保证输入变量中的一个变化可以尽可能快地输出为一个脉冲

（续）

序号	参数	数据类型	数值范围	默认	说　　明
11	COM_RST	BOOL		FALSE	COMPLETE RESTART（完全再起动） 该块有一个初始化程序，可以在输入参数 COM_RST 置位时运行
12	CYCLE	TIME	≥1ms	T#10ms	SAMPLING TIME（采样时间） 块调用之间的时间必须恒定。该输入参数规定了块调用之间的时间

输入参数的数值在块中没有限制。没有参数检查。

4）输出参数：SFB 43/FB 43 “PULSEGEN” 输出参数的说明见表 6-22。

表 6-22　SFB 43/FB 43 “PULSEGEN” 输出参数的说明

序号	参数	数据类型	数值范围	默认	说　　明
1	QPOS_P	BOOL		FALSE	OUTPUT POSITIVE PULSE（输出正脉冲） 如果有脉冲输出，输出参数“输出正脉冲”被置位。在三级调节中总是正脉冲输出。在两级调节中，QNEG_P 总是与 QPOS_P 反向
2	QNEG_P	BOOL		FALSE	OUTPUT NEGATIVE PULSE（输出负脉冲） 如果有脉冲输出，输出参数“输出负脉冲”被置位。在三级调节中总是负脉冲输出。在两级调节中，QNEG_P 总是与 QPOS_P 反向

七、功能块举例

可以从网上下载。项目由几个具有不同复杂程度和目标点的注释 S7 程序组成。

http：//www4. ad. siemens. de/WW/view/en/16532187

第七章　S7-300/400的通信及网络

随着计算机网络通信技术的发展，自动控制方式由传统的集中式控制向多级分布式方向发展。为了适应这种形势的发展，世界各 PLC 生产厂家纷纷给自己的产品增加了通信及联网的功能，并研制开发出各自的 PLC 网络系统。如三菱的 MELSEC. NET，欧姆龙的 SYSMAC，西门子的 SINEC H3 局域网等。现在微型和小型的 PLC 也都具有了网络通信接口等功能。今后网络总的发展趋势是向高速、多层次、大信息量、高可靠性和开放式（即通信协议向国际标准或地区通用工艺标准靠近）的方向发展。

本章先介绍一些通信网络的基础知识，在此基础上主要介绍西门子 S7-300/400 PLC 的通信和网络技术。

第一节　通信及网络基础

在实际工作中，无论是计算机之间还是计算机的 CPU 与外部设备之间常常要进行数据交换。不同的独立系统由传输线路互相交换数据便是通信，构成整个通信的线路称为网络。通信的独立系统可以是计算机、PLC 或其他有数据通信功能的数字设备，称为数据终端（Data Terminal Equipment，DTE）。传输线路的介质可以是双绞线、同轴电缆、光纤或无线电波等。

一、数据通信方式

1. 数据传输方式

（1）按照传输数据的时空顺序，数据的通信可分为并行通信（Parallel Communication）和串行通信（Serial Communication）

1）并行通信：从概念上说，并行传输的机制很简单，即一次使用 n 条导线来传输 n 个比特。这种方式下，每个比特都使用专用的线路，如图 7-1 所示，而一组中的 n 个比特就可以在每个时钟脉冲从一个设备传输到另一个设备。通常八根导线被捆成一根电缆，两端都有连接头。

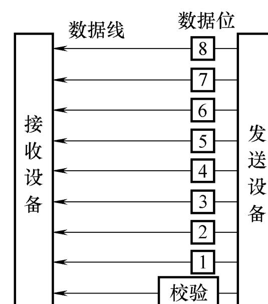


图 7-1　并行数据传输

并行传输的优势在于速度。当其他因素相同时，并行传输将比串行传输的速度快 n 倍，但同时也存在一个严重缺点——费用高。为进行数据传输，并行传输需要 M 条通信线路（在本例中是导线）。因为比较昂贵，所以并行传输通常被限制在最长 25ft（1ft = 0.3048m）的距离内。而且在传输过程中，容易因线路因素使电压电平发生变化，最常见的是电压衰减和信号互相干扰问题（Cross Talk），使得传输的数据发生错误。

2）串行通信：在串行传输中，比特是一个一个依次发送的。因此在两个通信设备之间传输数据只需要一条通信信道，而不是 n 条。串行传输如图 7-2 所示。由于串行传输只需要一条通信信道，因此其费用大约只是并行传输的 n 分之一。因为在设备内部的传输是并行的，所以在发送端和线路之间以及接收端和线路之间的接口上，都需要有转换器（前者是并/串转换，后者是串/并

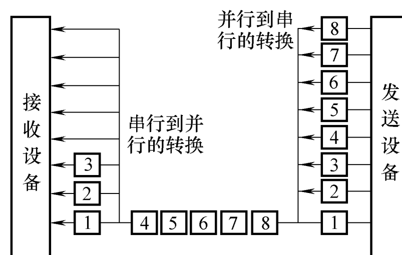


图 7-2　串行传输

转换)。

如果通信距离小于 30m 可采用并行通信，例如计算机（PC）与打印机的通信，PLC 的内部各元件之间、主机与扩展模块之间。当距离大于 30m 则要采用串行通信方式，例如计算机之间、计算机与 PLC 之间、PLC 和 PLC 之间等。

(2) 同步通信和异步通信

由传输介质连接起来进行数据交换的两台设备，其工作必须密切配合。每次经过介质发送的各位数据，其定时对发送机和接收机必须是相同的，即相互是同步的。因此需要考虑速率、持续时间和间隔之间的关系。同步就是接收端要按照发送端所发送的每个码元的起止时刻和重复频率来接收数据。两者时间上必须取得一致。传输数据的同步方式有：异步传输和同步传输两种方式。

1）异步通信异步传输（Asynchronous Transmission）：异步传送也称起止式传送，它是利用起止法来达到收发同步的。

异步传输将比特分成小组进行传送，小组可以是 8 位的 1 个字符或更长。发送方可以在任何时刻发送这些比特组，而接收方不知道它们会在什么时候到达。一个常见的例子是计算机键盘与主机的通信。按下一个字母键、数字键或特殊字符键，就发送一个 8 比特的 ASCII 代码。键盘可以在任何时刻发送代码，这取决于用户的输入速度，内部的硬件必须能够在任何时刻接收一个键入的字符。

异步传输存在一个潜在的问题，即接收方并不知道数据会在什么时候到达。在它检测到数据并做出响应之前，第一个比特已经过去了。这就像有人出乎意料地从后面走上来跟你说话，而你没反应过来，漏掉了最前面的几个词。因此，每次异步传输的信息都以一个起始位开头，它通知接收方数据已经到达了，这就给了接收方响应、接收和缓存数据比特的时间；在传输结束时，一个停止位表示该次传输信息的终止。按照惯例，空闲（没有传送数据）的线路实际携带着一个代表二进制 1 的信号，异步传输的开始位使信号变成 0，其他的位信号随传输的数据信息而变化。最后，停止位使信号重新变回 1，该信号一直保持到下一个开始位到达。例如在键盘上数字“1”，按照 8 比特的扩展 ASCII 编码，将发送“00110001”，同时需要在 8 比特的前面加一个起始位，后面一个停止位。

异步传输的实现比较容易，由于每个信息都加上了“同步”信息，因此计时的漂移不会产生大的积累，但却产生了较多的开销。在上面的例子中，每 8 个比特要多传送两个比特，总的传输负载就增加 25%。对于数据传输量很小的低速设备来说问题不大，但对于那些数据传输量很大的高速设备来说，25% 的负载增值就相当严重了。因此，异步传输常用于低速设备。

2）同步传送（Synchronous Transmission）：同步传送的比特分组要大得多。它不是独立地发送每个字符，每个字符都有自己的开始位和停止位，而是把它们组合起来一起发送。我们将这些组合称为数据帧，或简称为帧。

数据帧的第一部分包含一组同步字符，它是一个独特的比特组合，类似于前面提到的起始位，用于通知接收方一个帧已经到达，但它同时还能确保接收方的采样速度和比特的到达速度保持一致，使收发双方进入同步。

帧的最后一部分是一个帧结束标记。与同步字符一样，它也是一个独特的比特串，类似于前面提到的停止位，用于表示在下一帧开始之前没有别的即将到达的数据。

同步传输通常要比异步传输快得多。接收方不必对每个字符进行开始和停止操作。一旦检测到帧同步字符，它就在接下来的数据到达时接收它们。另外，同步传输的开销也比较少。例如，

一个典型的帧可能有500字节（即4000比特）的数据，其中可能只包含100比特的开销。这时，增加的比特使传输的比特总数增加2.5%，这比异步传输中25%的增值要小得多。随着数据帧中实际数据比特位的增加，开销比特所占的百分比将相应地减少。但是，数据比特越长，缓存数据所需要的缓冲区也越大，这就限制了一个帧的大小。另外，帧越大，它占据传输媒体的连续时间也越长。同步与异步传输的区别如下。

① 异步传输是面向字符的传输，而同步传输是面向比特的传输。

② 异步传输的单位是字符，而同步传输的单位是帧。

③ 异步传输通过字符起止的开始和停止码抓住再同步的机会，而同步传输则是从数据中抽取同步信息。

④ 异步传输对时序的要求较低，同步传输往往通过特定的时钟线路协调时序。

⑤ 异步传输相对于同步传输效率较低。

2. 数据传送方向

按串行通信的数据在通信线路进行传送的方向可分为单工、半双工和全双工通信方式三种。

(1) 单工通信方式

单工通信就是指数据的传送始终保持同一个方向，而不能进行反向传送，如图7-3所示。其中，A端只能作为发送端发送数据，B端只能作为接收端接收数据。

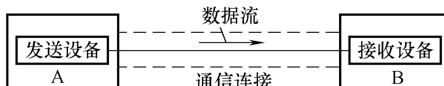


图7-3　单工通信方式

(2) 半双工通信方式

半双工通信就是指信息流可以在两个方向上传送，但同一时刻只限于一个方向传送，如图7-4所示。其中，A端和B端都具有发送和接收的功能，但传送线路只有一条，或者A端发送B端接收，或者B端发送A端接收。

(3) 全双工通信方式

全双工通信能在两个方向上同时发送和接收，如图7-5所示。A端和B端都可以一方面发送数据，一方面接收数据。

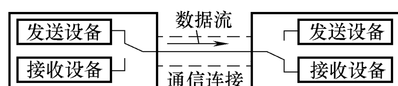


图7-4　半双工通信方式

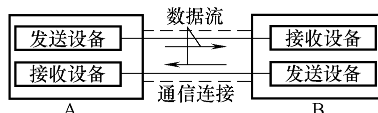


图7-5　全双工通信方式

二、信道和信道参数

1. 信道

信道是信号从发送端到接收端之间进行传输的路径。

这一术语也是线路、电路、链路、数据链路、通路等术语的同义语。这个术语实际上是指把两种数据通信设备（Data Communication Equipment，DCE）连接起来的设施，包括传输介质和有关中间通信设备。但信道概念的含义，并不是实际的电缆、电线等，而是指数据流过的由介质提供的路径。

2. 信道参数

作为信道组成的传输介质分有线和无线两类，这些传输介质有以下特性，即信息流方向、传送信号的类型、传输速率和带宽。

这里主要介绍传输速率和带宽。

1）带宽就是信道的通带（passband），由上、下限截止频率f_{c1}和f_{c2}来限定。一条传输信道上允许传输信息的频率范围，即能从信道上通过的最高信号频率，是传输系统的一个重要指标，系统所追求的带宽是系统应用的函数。

系统带宽由传输介质、接口部件、传输设备、传输协议及传输信息的特性等因素所决定。

2）传输速率：波特率，即数据传送速率，表示每秒钟传送二进制代码的位数，它的单位是 bit/s。假如数据传送速率是 120 字符/s，而每个字符包含 10 个代码位（一个起始位、一个终止位、8 个数据位），这时传送的波特率为

$$10\text{bit/字符} \times 120\ \text{字符/s} = 1200\text{bit/s}$$

三、传送介质

1. 双绞线（Twisted-Pair）

双绞线是现在最普通的传输介质，它由两条相互绝缘的铜线组成，典型直径为 1mm。两根线绞接在一起是为了防止其电磁感应在邻近线对中产生干扰信号。现行双绞线电缆中一般包含 4 个双绞线对，具体为橙 1/橙 2、蓝 4/蓝 5、绿 6/绿 3、棕 3/棕白 7。计算机网络使用 1-2、3-6 两组线对分别来发送和接收数据。双绞线接头为具有国际标准的 RJ-45 插头和插座。双绞线分为屏蔽（Shielded）双绞线（STP）和非屏蔽（Unshielded）双绞线（UTP），非屏蔽双绞线有线缆外皮作为屏蔽层，适用于网络流量不大的场合。屏蔽双绞线具有一个金属夹套（sheath），对电磁干扰（Electromagnetic Interference，EMI）具有较强的抵抗能力，适用于网络流量较大的高速网络协议应用。双绞线根据性能又可分为 5 类、6 类和 7 类，现在常用的为 5 类非屏蔽双绞线，其频率带宽为 100MHz，能够可靠地运行 4MB、ICME 和 16MB 的网络系统。当运行 100MB 以太网时，可使用屏蔽双绞线以提高网络在高速传输时的抗干扰特性。6 类、7 类双绞线分别可工作于 200MHz 和 600MHz 的频率带宽之上，且采用特殊设计的 RJ-45 插头（座）。值得注意的是，频率带宽（MHz）与线缆所传输数据的传输速率（Mbit/s）是有区别的——Mbit/s 衡量的是单位时间内线路传输的二进制位的数量，MHz 衡量的则是单位时间内线路中电信号的振荡次数。双绞线较多应用于基于载波感应多路访问/冲突检测（Carrier Sense Multiple Access/Collission Detection，CMSA/CD）技术，即 10BASE-T（10Mbit/s）和 100BASE-T（100Mbit/s）的以太网（Ethernet）中，具体规定如下。

1）一段双绞线的最大长度为 100m，只能连接一台计算机。

2）双绞线的每端需要一个 RJ-45 插件（头或座）。

3）各段双绞线通过集线器（Hub 的 10BASE-T 重发器）互连，利用双绞线最多可以连接 64 个站点到重发器（Repeater）。

4）10BASE-T 重发器可以利用收发器电缆连到以太网同轴电缆上。

2. 同轴电缆（Coaxial）

广泛使用的同轴电缆有两种：一种为 50Ω（指沿电缆导体各点的电磁电压对电流之比）同轴电缆，用于数字信号的传输，即基带同轴电缆；另一种为 75Ω 同轴电缆，用于宽带模拟信号的传输，即宽带同轴电缆。同轴电缆以单根铜导线为内芯，外裹一层绝缘材料，外覆密集网状导体，最外面是一层保护性塑料。金属屏蔽层能将磁场反射回中心导体，同时也使中心导体免受外界干扰，故同轴电缆比双绞线具有更高的带宽和更好的噪声抑制特性。

现行以太网同轴电缆的接法有两种，直径为 0.4cm 的 RG-11 粗缆采用凿孔接头接法，直径为 0.2cm 的 RG-58 细缆采用 T 形头接法。粗缆要符合 10BASE5 介质标准，使用时需要一个外接收发器和收发器电缆，单根最大标准长度为 500m，可靠性强，最多可接 100 台计算机，两台计算机的最小间距为 2.5m。细缆按 10BASE2 介质标准直接连到网卡的 T 形头连接器（即 BNC 连接

器）上，单段最大长度为 185m，最多可接 30 个工作站，最小站间距为 0.5m。

3. 光导纤维（Fiber Optic）

光导纤维是软而细的、利用内部全反射原理来传导光束的传输介质，有单模和多模之分。单模（模即 Mode，入射角）光纤多用于通信业。多模光纤多用于网络布线系统。

光纤为圆柱状，由 3 个同心部分组成——纤芯、包层和护套，每一路光纤包括两根，一根接收，一根发送。用光纤作为网络介质的 LAN 技术主要是光纤分布式数据接口（Fiber-optic Data Distributed Interface，FDDI）。与同轴电缆比较，光纤可提供极宽的频带且功率损耗小、传输距离长（2km 以上）、传输率高（可达数千 Mbit/s）、抗干扰性强（不会受到电子监听），是构建安全性网络的理想选择。

4. 微波传输和卫星传输

这两种传输方式均以空气为传输介质，以电磁波为传输载体，联网方式较为灵活。几种常见的传输介质性质比较见表 7-1。

表 7-1　传输介质性质比较

性能	传 送 介 质		
	双绞线	同轴电缆	光缆
传输速率	9.6k ~ 2Mbit/s	1 ~ 450Mbit/s	10 ~ 500Mbit/s
连接方法	点到点 多点 1.5km 不用中继器	点到点 多点 10km 不用中继器（宽带） 1 ~ 3km 不用中继器（基带）	点到点 50km 不用中继器
传输信号	数字、调制信号、纯模拟信号（基带）	调制信号，数字（基带），数字、声音、图像（宽带）	调制信号（基带）、数字、声音、图像（宽带）
支持网络	星形、环形、小型交换机	总线型、环形	总线型、环形
抗干扰	好（需外屏蔽）	很好	极好
抗恶劣环境	好	好，但必须将电缆与腐蚀物隔开	极好，耐高温和其他恶劣环境

四、网络传输设备

1. 中继器

中继器（Repeater）又译成重发器，是最简单的连接设备，工作于网络的物理层。它的作用是对网络电缆上传输的数据信号经过放大和整形后再发送到其他电缆段上。因此，中继器实际上只能算是数字信号的再生放大器。

经过中继器连接的两段电缆上的工作站就像是在一条加长了的电缆上工作一样，在一段电缆上的冲突也将被中继器传到另一段电缆上。用中继器扩展的网络，不管增加多大的距离范围，该网络在逻辑上和物理上都是同一个网络整体。

使用中继器时不能形成环路，并且要考虑到网络的传输延迟和负载情况，不能无限制的连接中继器。例如：以太网用粗同轴电缆联网，电缆段最大距离为 500m，细同轴电缆最大距离为 185m，采用中继器扩展网络，Ethernet 最多可用 4 个中继器。

中继器按其接口个数可分为双口中继器和多口中继器。前者有两个接口，一个用于输入，另一个用于输出；后者接口数大于两个，又称为集线器（Hub）。按连接的传输介质可分为：电缆中继器（用于双绞线、同轴电缆）和光缆中继器（用于连接光缆）。

2. 集线器

集线器（Hub）又称集中器，是多口的中继器。把它作为一个中心节点，可用它连接多条传

输介质。其优点是当某条传输介质发生故障，不会影响到其他的节点。

集线器分为有源集线器（Active Hub）、无源集线器（Passive Hub）和智能集线器。

（1）无源集线器

只是把相近地区的多段传输介质集中到一起，对它们所传输的信号不做任何处理，而且对它所集中的传输介质，只允许扩展到最大有效距离的一半。

（2）有源集线器

把相近地区的多段传输介质集中到一起，还对每条传输的电信号有整形、放大和转发作用，并具有扩展传输介质长度的功能。

（3）智能集线器

具备有源集线器的功能，还具有网络管理、路径选择等功能。随着网络技术的发展，现在集线器发展成为交换集线器（Switching Hub）。交换集线器不但能使网络分段，并增加了线路交换功能，提高了传输带宽。

3. 网桥

网桥是用于连接两个或两个以上具有相同通信协议、传输介质及寻址结构的局域网间的互联设备，它工作于网络的数据链路层。网桥有它的软件和硬件，因此，通过网桥的数据帧，不需要进行网络协议转换及原语连接的转换，只需要附加新的路径信息和全局地址。网桥需要有足够大的 RAM 缓冲区，用于扩展网络距离和转发数据到另一个目的网工作站。若缓冲区溢出，则会把发送的帧丢掉，不做任何处理，而是等待端到端的协议处理。

但是，网桥对广播信息不能识别，也不能过滤。于是容易产生 A 网络广播给 B 网络工作站数据，又被重新广播回 A 网络，这种往返广播，使网上出现大量冗余信息，最终形成广播风暴。

网桥分为本地网桥和远程网桥。本地网桥指所连接的两个 LAN 间的距离在 LAN 所允许的最大传输介质长度之内的网桥。远程网桥则反之。远程网桥连接两个 LAN 时，必须使用调制解调器，所以在连接两个远距离的 LAN 时，需要两个网桥。而本地网桥只需用一个网桥去连接两个 LAN 或远程工作站。网桥按功能可以分为：透明桥、源路由桥、翻译桥和打包桥四种。

4. 路由器

按连接网络的网络层协议，路由器分为单协议路由器（只能对具有相同网络层协议的网络互连）和多协议路由器（包括多种网络层协议）。按网络互连距离可分为远程路由器和本地路由器等。

路由器工作于网络层，路由器的主要功能有：

1）选择最佳的转发数据的路径，建立非常灵活的连接，均衡网络负载。

2）利用通信协议本身的流量控制功能来控制数据传输，有效地解决拥挤问题。

3）具有判断需要转发的数据分组的功能，不仅可根据 LAN 网络地址和协议类型，而且可根据网间地址、主机地址、数据类型（如文件传输、远程登录或电子邮件）等，判断分组是否应该转发。对于不该转发的信息（包括错误信息），都过滤掉，从而可避免广播风暴，比网桥有更强的隔离作用，提高安全保密性能。

4）把一个大的网络划分为若干个子网。

5. 网关

网关（Gateway），又称高层协议转发器，一般用于不同类型且差别较大的网络系统间的互联，又可用于同一个物理网而在逻辑上不同的网络间互连。

第二节 通信网络结构

一、网络概述

将具有独立功能而又分散在不同地理位置的多台计算机，通过通信设备和通信线路连接起来构成的计算机系统称为计算机网络。PLC 与计算机之间或多台 PLC 之间也可直接或通过通信处理器构成网络，以实现信息交换；各 PLC 或远程 I/O 模块按功能各自放置在生产现场进行分散控制，再用网络连接起来，组成集中管理的分布式网络。分布式网络以适应性强、扩展性好及维护简单等优势而得到广泛应用。互连和通信是网络的核心，网络的拓扑结构、传输控制、传输介质和通道利用方式是构成网络的四大要素。

二、网络体系结构——IEEE802 参考模型和 ISO 标准

1. IEEE802.1 参考模型

自 1980 年以来，许多国家和国际标准化机构都在积极进行局域网的标准化工作，其中最有影响力的是 IEEE 制定的局域网的 802 标准，包括 CSMA/CD、令牌总线和令牌环等，它被 ANSI 接受为美国国家标准，被 ISO 作为国际标准（称为 ISO8802 标准）。这些标准在物理层和 MAC 子层上有所不同，但在数据链路层上是兼容的。

IEEE802 的 LAN 标准遵循 OSI 参考模型的分层原则，描述最低两层——物理层和数据链路层的功能以及与网络层的接口服务，其中数据链路层又分成介质访问控制子层（MAC）和逻辑链路控制子层（LLC）。

IEEE802 参考模型及其与 OSI 参考模型对比关系如图 7-6 所示。

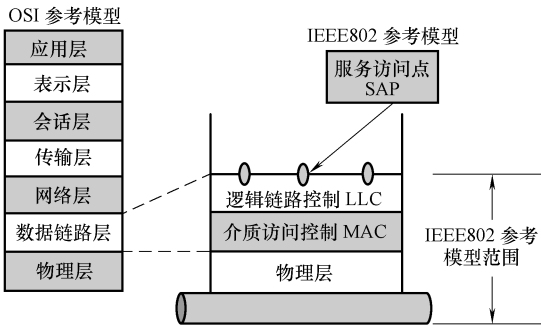


图 7-6 IEEE802 参考模型及其与 OSI 参考模型对比关系

IEEE802.1 标准规定局域网的低三层的功能如下。

（1）物理层

与 OSI/RM 的物理层相对应，但所采用的具体协议标准的内容直接与传输介质有关。

（2）介质访问控制层

MAC 层具体管理通信实体接入信道而建立数据链路的控制过程。

（3）逻辑链路控制层

LLC 层提供一个或多个服务访问点，以复用的形式建立多点——多点之间的数据通信连接，并包括寻址、差错控制、顺序控制和流量控制等功能。这些功能基本上与 HDLC 规程一致。此外，在 LLC 层还提供本属于 OSI/RM 中网络层提供的两项服务，即无连接的数据报服务和面向连接的虚电路服务。

由图 7-6 可见，MAC 层和 LLC 层合并在一起，近似等效于 OSI 参考模型中的数据链路层。LLC 层协议与局域网的拓扑形式和传输介质的类型无关，它对各种不同类型的局域网都是适用的。然而，MAC 层协议却与网络的拓扑形式及传输介质的类型直接相关，其主要作用是介质访问控制和对信道资源的分配。例如，总线型局域网主要采用竞争式的随机访问控制协议，最典型的是 CSMA/CD，还有令牌总线、令牌环等标准。

2. OSI 标准各层

（1）物理层

物理层是 OSI 模型的最低层，它向下直接与传输介质相连接，向上相邻且服务于数据链路层。它的作用是在数据链路层实体之间提供必需的物理连接，按顺序传输数据比特，并进行差错检查。在发现错误时，向数据链路层提交报告。物理层重点考虑的是怎样才能在连接各种计算机的传输介质上传输数据的比特流，而不是连接计算机的具体的物理设备或具体的传输介质。

具体来说，物理层协议要解决的是主机、工作站等数据终端设备与通信线路上通信设备之间的接口问题。按照国际标准化组织（ISO）的术语，将这两种设备分别称为 DTE 和 DCE。

DTE（Data Terminal Equipment）又称数据终端设备，指数据输入、输出设备和传输控制器或者计算机等数据处理装置及其通信控制器。DCE（Data Circuit-Terminating Equipment）又称数据电路端接设备，指自动呼叫设备、调制解调器（Modem）以及其他一些中间装置的集合。DTE 的基本功能是产生、处理数据；DCE 的基本功能是沿传输介质发送和接收数据。物理层信号线如图 7-7 所示。



图 7-7　物理层信号线

物理层具有以下 4 个基本特性。

1）机械特性：机械特性规定了 DTE 与 DCE 实际的物理连接。DTE 和 DCE 作为两种分立设备，通常采用接插件实现机械上的互连。机械特性详细说明了接插件的形状和尺寸、插头的数目、排列方式以及插头和插座的尺寸、电缆的长度以及所含导线的数目等。这很像平时常见的各种规格的电源插头的尺寸都有严格的规定。ISO 物理层的机械特性的标准有 ISO 2110、ISO 2593、ISO 4092 和 ISO 4093。

2）电气特性：电气特性规定了数据交换信号以及有关电路的特性。一般包括最大数据传输率的说明、表示信号状态（逻辑电平，通/断，传号/空号）的电压和电流的识别，即什么样的电压表示 1 或 0，以及电路特性的说明和与互联线路相关的规定。ISO 物理层采用的电气特性的标准有 CCITT V. 10/X. 26、CCITT V. 11/X. 27、CCITT V. 28 和 CCITT V. 35。

3）功能特性：说明某条线上出现的某一电平的电压表示何种意义。即每一条线的功能分配和确切定义。ISO 物理层采用的功能特性标准有 CCITT V. 24 和 CCITT X. 24。通常信号线可分为数据线、控制线、同步线和地线。

4）规程特性：即通信协议，说明对于不同功能的各种可能事件的出现顺序。即各信号线的工作规则和先后顺序。ISO 物理层采用的规程特性标准有 CCITT X. 20/21/22 和 CCITT V. 24/25。

（2）数据链路层

链路就是一条无源的点到点的物理线路段，中间没有任何其他的交换节点。

数据链路则是另一个概念。这是因为当需要在一条线路上传输数据时，除了必须有一条物理线路外，还必须有一些必要的规程来控制这些数据的传输。把实现这些规程的硬件和软件加到链路上，就构成了数据链路。

物理层以比特为单位进行数据传输，数据链路层以帧为单位进行数据传输。

帧是具有一定长度和格式的信息块，一般由一些字段和标志组成。不同网络其帧格式或长度可以不同，但将比特流分成帧的方法基本相同。四种常用的方法为：字符计数法；带填充字符的首尾界符法；带填充比特的首尾标志法；物理层编码违例法。

把比特流分成帧，标定帧的起始和结束，以利于进行差错控制。在数据链路层，数据的传送单位是帧。数据一帧一帧地传送，就可以在出现差错时，将有差错的帧重传一次，而避免将所有

数据重传，从而实现差错控制。

数据链路层协议有时也称为通信控制规程。它是为了实现传输控制所制定的一些规格和顺序。通用性比较强的计算机网络通信规程主要有美国国家标准协会（ANSI）提出的高级数据通信控制规程 ADCCP；国际标准化组织（ISO）推荐的高级数据链路控制规程 HDLC；国际电报电话咨询委员会（CCITT）发表的 X.25 建议书。

通信控制规程归纳起来可分为面向字符型和面向比特型。

所谓面向字符就是在链路上所传送的数据必须是由规定字符集（例如 ASCII）中的字符所组成。在链路上传送的控制信息也必须由同一个字符集中的若干指定的控制字符构成。最典型的面向字符的协议是 IBM 的 BSC 协议（二进制同步通信）。

在面向比特型的通信控制规程中，报文的数据和控制信息完全独立，具有良好的透明性；差错检验一般采用纠错码方式，可靠性强；在链路上可进行信息连接和双向发送，传输效率高；信息传输都统一以帧为单位，控制简单。

（3）网络层

网络层是 OSI 参考模型中的第三层，它建立在数据链路层所提供的两个相邻端点之间的数据帧的传送功能之上，将数据从源端经过若干中间节点传送到目的端。从而向传输层提供最基本的端到端的数据传送服务。网络层是处理端到端数据传输的最底层，体现了网络应用环境中资源子网访问通信子网的方式。

网络层主要完成以下几方面的功能。

1）路由控制：利用网络拓扑结构等网络状态，选择分组传送路径。这是网络层的主要功能。在大多数子网中，分组的整个过程需要经过多次转发。无线广播网络是唯一的例外。

2）拥塞控制：控制和预防网络中出现过多的分组。当到达通信子网中某一部分的分组数高于一定的阈值，使得该部分网络来不及处理这些分组时，就会使这部分以至整个网络的性能下降。这种情况称为拥塞。网络拥塞带来的影响如图 7-8 所示。

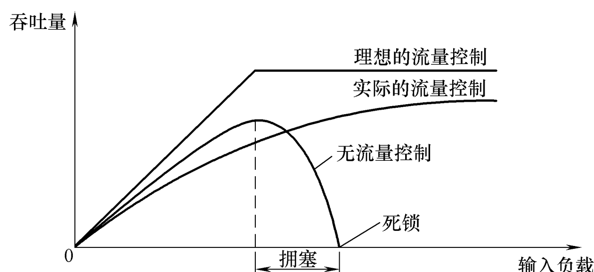


图 7-8　网络拥塞带来的影响

3）透明传输：透明传输就是不管所传数据是什么样的比特组合，都应当能够在链路上传送。当所传数据中的比特组合恰巧出现了与某一个控制信息完全一样时，必须采取适当的措施，使收方不会将这样的数据误认为是某种控制信息。这样才能保证数据链路层的传输的透明。

4）异种网络的互联：解决不同网络在寻址、分组大小、协议等方面的差异。不同类型的网络对分组大小、分组结构等的要求都不相同，因此要求在不同种类的网络交界处的路由器能够对分组进行处理，使得分组能够在不同网络上传输。

5）分组生成和装配：传输层报文与网络层分组间的相互转换。传输层报文通常很长，不适合直接在分组交换网络中传输。在发送端，网络层负责将传输层报文拆成一个个分组，再进行传输。在接收端，网络层负责将分组组装成报文交给运输层处理。

（4）传输层

传输层是利用网络层的服务和传输实体的功能，向会话层提供服务。传输层是整个协议层次结构的核心。其任务是为从源端机到目的机提供可靠的、价格合理的数据传输，而与当前网络或使用

的网络无关。如果没有传输层，整个分层协议的概念也就没有什么意义了。传输层地位关系图如图 7-9 所示。

传输层处于 OSI 模型的上 3 层与下 3 层之间，提供进程间端到端的、透明的、可靠的数据传送。网络层提供系统间的数据传送，而传输层提供进程间的数据传送。

OSI 模型中上 3 层的功能为信息传送，了解数据含义，进程间通信；下 3 层的功能为数据传送，不关心数据含义以及系统间通信。

传输层可视为低层的一部分，可弥补高层（上 3 层）要求与网络层（基于下 3 层）数据传送服务质量间的差异（差错率、差错恢复能力、吞吐率、延时、费用等），对高层屏蔽网络层的服务的差异，提供稳定和一致的界面，其抽象模型如图 7-10 所示。

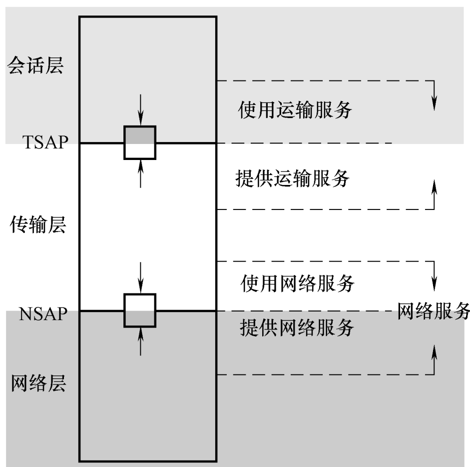


图 7-9 传输层地位关系图

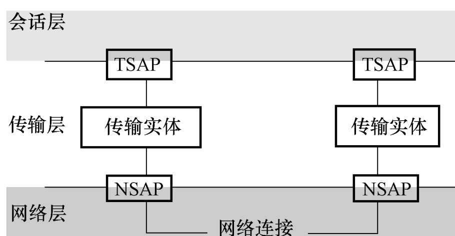


图 7-10 传输层抽象模型

（5）会话层

会话层位于 OSI 参考模型的第五层，它是面向信息处理的 OSI 高层和面向数据通信的 OSI 低层的接口。会话协议的最主要的目的是提供一个面向用户的连接服务，会话层给用户间的对话和活动提供组织和同步所必需的手段，对数据传送提供控制和管理。虽然传输层能负责端到端的可靠通信服务，但仍不能满足许多应用的需求，会话层的设立可以做到为传输层“增值”的功能，以便提供一个面向应用的完善的服务。

具体说来会话层有以下几个功能。

1）会话管理：连接建立、数据传送、连接释放。

2）令牌管理：管理和协商数据传送、同步及对话连接的释放时必需的发送权（即 Token），设定半双工及全双工的数据传送模式。

3）同步管理：在连续的数据传送过程中插入同步点，当出错时，可以从双方认为合适的同步点开始重新传送。

4）活动管理：将报文流分成活动（activity）逻辑单元，对用户应用的交互活动过程进行结构化管理，即每一个活动独立于其前、后到达的活动。

5）异常情况的处理：在会话期间报告来自下面网络的异常情况，保证在会话连接释放之前所有的数据单元都被应用进程所接收。

（6）表示层

表示层主要解决的就是信息以什么样的表现形式（数据表现）传送给对方。不关心处理的用户数据有什么样的意义，只考虑用什么样的传送形式传送这一问题，也就是说表示层的功能并不是信息的具体表达，而是处理信息表示中所遇到的问题，考虑如何将不同信息的表达形式转换成公共的信息传送形式。表示层模型如图 7-11 所示。

OSI 模型中，信息的表示涉及语法和语义两个方面。

1）语义：数据的内容与意义，由应用层的各应用协议处理。表示层不关心数据具体的语

义，只有应用实体才能知道数据的意义。例如，文件的记录组成、作业的执行方法、终端的画面控制等与意义内容有关的方面。

2）语法：数据的表示形式，由表示层处理。它解决异种计算机系统之间的信息表示形式的差异。例如，文字、图形、声音的表示，数据压缩、数据加密等与表示形式有关的方面。语义和语法的关系可以举一个例子来说明：FLOWER，语义上是花的意思，语法上我们将它看成是字符串。

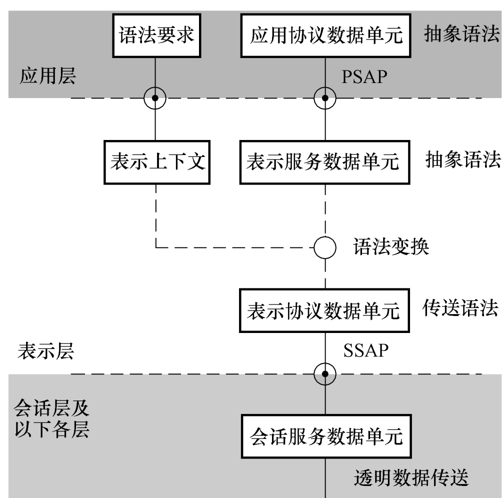


图 7-11　表示层模型

（7）应用层

应用层是 OSI/RM 中最高的一个功能层，它是开放系统互连环境（OSI 环境）与本地系统的操作系统和应用系统直接接口的一个层次。在功能上，应用层为本地系统的应用进程（AP）访问 OSI 环境提供手段，也是唯一直接给应用进程提供各种应用服务的层次。根据分层原则，应用层向应用进程提供的服务是 OSI 的所有层直接或间接提供服务的总和。

计算机通信网的最终目的是为用户提供一些特定的服务，使得本地系统能与外界系统进行协调合作。为了实现这种协调，应用层一方面为应用进程提供彼此通信的手段，也就是为其创建 OSI 环境；另一方面，由于各种应用类型的多样性，应用层协议也必定是多种多样的，为了减少应用系统与外界通信的复杂性，在应用层内应配置尽可能多的、公用或专用的应用服务元素（Application Service Element，ASE），提供给应用系统根据需要调用。所谓应用服务元素就是各种应用都需要的功能成分，是应用层的基本构件。

不同的应用协议可以采用相同的低层通信协议，实际上，应用进程之间的通信问题在传输层就已经基本解决了。至于在传输层上增加会话层和表示层这两个层次，是因为 OSI 的设计者认识到不同类型应用的应用进程在相互通信时表现出许多相似的特征，把这些相似的特征提取出来，分别设立会话层和表示层，这样就可以简化应用进程的设计和实现。

三、数据通信的网络拓扑结构

1. 网络拓扑结构

在网络中，通过传输线路互连的点称为节点，节点也可定义为网络中通向任何一个分支的端点，或通向两个或两个以上分支的公共点。各个节点互连的方式和型式称为网络拓扑。常用的拓扑结构有树形、星形、总线型和环形等，如图 7-12 所示。

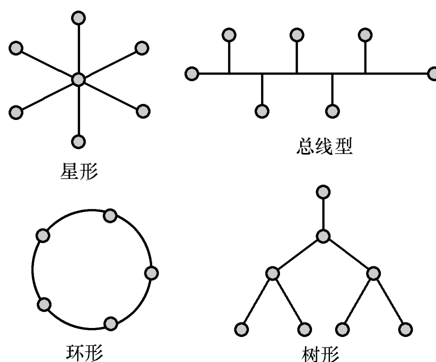


图 7-12　网络拓扑结构

（1）树形结构

树形结构在分级分布通信系统中广泛使用。其特点是，通信控制软件比较简单，而且为控制和差错处理提供了一个集中点。结构中处于较高位置的站点控制位于它下面的那些站点的数据通信。同级站点的数据传输要通过上一级站点的转接来实现。当某一级站点发生故障时，它下级站点的通信就会瘫痪，而它上级站点的通信仍能进行，只是不能与该站点进行通

信。若一个上级站点连接的下级站点较多，数据通信量较大时。会发生“瓶颈阻塞”问题。

（2）总线型结构

总线型结构是利用总线把所有节点连接起来，其特点是所有站点共享一个公共通信总线，总线不封闭，容易挂接新的站点或摘除旧站点。在主干链路（总线）上，任何时刻只允许两站间进行通信，两个节点间通过总线直接通信，速度快，延迟开销小。某站点发生故障时，对整个系统的影响较小。通信协议简单，但有时会出现争用总线控制权，而降低传输效率的问题。通信总线一旦发生故障，整个通信系统就会瘫痪。为了解决这个问题，通常采用冗余总线。通信介质常使用双绞线、同轴电缆或光纤，特别适用于工业控制领域，是工业控制局域网中常用的拓扑结构。

（3）星形结构

星形结构是以中央节点为中心与各个节点连接组成，网络中任何两个节点要进行通信都由中央控制站点控制并转换。其特点是：结构简单、控制容易、数据流向明确、便于程序集中开发和资源共享。但由于中央控制站点负责整个系统的数据交换，存在着“瓶颈阻塞”和“危险集中”两大问题。如果采用冗余中央控制站点的方法来解决，则会增加系统的复杂程度和成本。在小系统、通信不频繁的场合可以使用。上位计算机（也称主机、监控计算机、中央处理机）通过点到点的方式与各现场处理器（也称从机、下位机）进行通信就是一种星形结构。星形结构可使用双绞线为传输介质。

（4）环形结构

环形结构是环形网中各个节点首尾顺序连接形成。一个环在多数情况下，信息是以一个方向在环上从源点传递到目的点。每个站都是通过一个中继器连接到网络上，数据以分组形式发送，由于多节点共用一条环线，需对此进行控制，以决定每个站什么时候可把信息放在环上发送，每个节点都有控制发送和接收的访问逻辑，环形网络常用传输速率高达 10Mbit/s 的双绞线作为传输介质。其特点是：结构简单、挂接或摘除节点容易、安装费用低。每个站点的任务就是接收相邻上一站点发送的数据，然后把数据发送到相邻的下一个站点上。各站点之间可采用不同的传输媒体和不同的传输速度。在数据通信频繁的场合，它的传输效率比较高。但某个站点发生故障会阻塞信息通路，可靠性较差，节点较多时会影响传输效率。

2. 介质访问控制

介质访问控制也称传输控制，是指对网络通道占有权的管理和控制。常用的有 CSMA/CD、Token Bus、Token Ring 等。

（1）CSMA/CD

CSMA/CD 是一种总线争用协议，它一般用于总线网，每个站都能独立地决定帧的发送。如两个站或多个站同时发送，即产生冲突，同时发送的所有帧都会出错。每个站必须有能力判断冲突是否发生，如发生冲突，则应等待随机时间间隔后重发，以避免再次发生冲突。

（2）Token Ring

令牌环由一组用传输介质串联成一个环的站组成，环中有一个令牌在循环传送。任何一个站要发送数据，都必须等待循环的令牌通过该站。令牌是一种特殊的位组合，是一种发送权标志，如其形式可为 01111111，表示空令牌。希望发送数据帧的站等到空令牌来后，将空令牌改成忙令牌，其形式为 01111110，然后紧接着传送数据帧。数据信息一个比特接一个比特地附加到环上，环上信息从一站到下一站地环行。所寻址的目的站在信息经过时复制此信息，最后由发送该信息的站从环上撤除此信息，并将忙令牌改为空令牌，传至后面的站，使之获得发送权。

（3）Token Bus

CSMA/CD 介质访问控制采用总线争用方式，具有结构简单、在轻负载下延迟小等优点，随着负载的增加，冲突概率增加，性能明显下降；令牌环访问控制在重负载下利用率高，性能对传输距离不敏感且公平访问，但环形网结构复杂并存在检错能力和可靠性差等问题。令牌总线介质访问控制（Token Bus）是在综合上面两种介质访问控制的优点的基础上形成的一种介质访问控制方法。

四、现场总线

在传统的自动化工厂中，位于生产现场的许多设备和装置，当这些装置和设备相距较远、分布较广时，人们迫切需要一种可靠、快速、能经受工业现场环境的价格低廉的通信总线，将分散于现场的各种设备连接起来，对其实施监控。现场总线（Field Bus）就是在这样的背景下产生的。

现场总线是“安装在过程区域的现场设备/仪表与控制室内的自动控制装置/系统之间的一种串行、数字式、多点通信的数据总线”。

1. 现场总线的特点

现场总线具有以下几个方面的特点。

1）信息集成度高。现场总线可以从现场设备中获取大量丰富的信息，它不单纯取代 4～20mA 信号，还可以实现现场设备状态、故障参数信息的传递。系统除了完成远程控制，还可以完成远程参数化工作。现场总线能够很好地满足工厂自动化、CIMS 系统的信息集成要求，实现办公自动化与工业自动化的紧密结合，形成新型的管控一体化的全开放工业控制网络。

2）开放性、互操作性、互换性、可集成性。不同厂家的产品，只要使用同一总线标准，就具有互操作性、互换性，设备具有很好的可集成性。系统为开放式，允许其他厂商将自己专长的控制技术，如控制算法、工艺流程、配方等集成到通用的系统中。

3）系统的可靠性高、维护性能好。现场总线采用总线方式替代二对一的 I/O 连接，对于大规模的 I/O 系统来说，减少了由于接线点造成的不可靠因素。同时，系统具有现场设备的在线故障诊断、报警、记录功能，可完成现场设备的远程参数设定、修改等参数化工作，也增强了系统的可维护性。

4）实时性好、成本低。现场总线处于通信网路的最底层，完成具体的生产和协调任务，因而实时性好，造价相对低廉。对大范围、大规模的 I/O 分布式系统来说，省去了大量的电缆、I/O 模块及电缆铺设的工程费用，降低了系统和工程的成本。

现场总线是现场设备互连的最有效手段，它能最大限度地发挥和调度现场及设备的智能处理功能，它在控制设备和传感器之间提供给用户的双向通信能力是以往任何体系机构都无法提供和不可匹敌的。现场总线以开放的、独立的、全数字化的双向多变量通信代替 0～10mA 或 4～20mA 现场仪表信号。现场总线 I/O 集检测、数据处理、通信为一体，可以代替变送器、调节器、记录仪等模拟仪表，使用现场总线后，自控系统的配线、安装、调试和维护等方面的费用可以大大节省。

2. 现场总线的国际化标准

由于历史的原因，现在有多种现场总线标准并存，IEC 的现场总线国际标准（IEC 61158）是迄今为止制订时间最长、意见分歧最大的国际标准之一。它的制订时间超过 12 年，先后经过 9 次投票，在 1999 年底获得通过。经过多方的争执和妥协，最后容纳了 8 种互不兼容的协议，这 8 种协议在 IEC 61158 中分别为 8 种现场总线类型，即：

类型 1：原 IEC 61158 技术报告，即现场总线基金会（FF）的 H1；

类型 2：Control Net（美国 Rockwell 公司支持）；

类型 3：PROFIBUS（德国西门子公司支持）；

类型 4：P—Net（丹麦 Process Data 公司支持）；

类型 5：FF 的 HSE（原 FF 的 H2，高速以太网，美国 Fisher Rosemount 公司支持）；

类型 6：Swift Net（美国波音公司支持）；

类型 7：World FIP（法国 Alstom 公司支持）；

类型 8：Interbus（德国的 Phoenix contact 公司支持）。

各类型将自己的行规纳入 IEC 61158，且遵循两个原则。

1）不改变 IEC 61158 技术报告的内容。

2）不改变各行规的技术内容，各组织按 IEC 技术报告（类型 1）的框架组织各自的行规，并提供对类型 1 的网关或链接器。用户在使用各种类型时仍需使用各自的行规。因此 IEC 61158 标准不能完全代替各行规，除非今后出现完整的现场总线标准。

IEC 标准的 8 种类型都是平等的，类型 2～8 都对类型 1 提供接口，标准并不要求类型 2～8 之间提供接口。

第三节　S7-300/400 的通信网络

一、工业自动化网络

现代大型工业企业中，一般采用多级网络的形式。PLC 制造商经常用生产金字塔结构来描述其产品可实现的功能。这种金字塔结构的特点是：上层负责生产管理，底层负责现场监测与控制，中间层负责生产过程的监控与优化。国际标准化组织（ISO）对企业自动化系统确立了初步的模型，如图 7-13 所示。

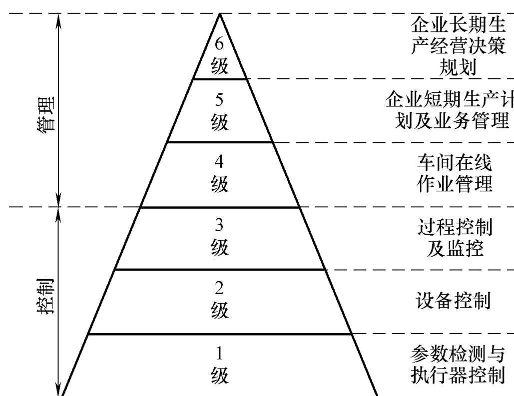


图 7-13　自动化系统模型

工厂自动化系统中，不同生产厂家的网络结构的层数及各层的功能分布有所差异。但基本上都是由从上到下的各层在通信基础上相互协调，共同发挥着作用。实际工厂中一般采用 2～4 级子网构成复合型结构，而不一定是这 6 级，各层应采用相应的通信协议。

下半部分的控制部分包括参数检测及执行、设备控制、过程控制及监控对应着实际的现场设备层、单元层和工厂管理层。

（1）现场设备层

执行器—传感器的主要功能是连接现场设备，例如分布式 I/O、传感器、驱动器、执行机构和开关设备等，完成现场设备控制及设备间联锁控制。主站（PLC、PC 或其他控制器）负责总线通信管理及与从站的通信。总线上所有设备生产工艺控制程序存储在主站中，并由主站执行。

（2）单元层

单元层又称为车间监控层，用来完成车间主生产设备之间的连接，实现车间级设备的监控。车间级监控包括生产设备状态的在线监控、设备故障报警及维护等。通常还具有诸如生产统计、生产调度等车间级生产管理功能。车间级监控通常要设立车间监控室，有操作员工作站及打印设备。车间级监控网络可采用 PROFIBUS-FMS 或工业以太网，PROFIBUS-FMS 是一个多主网络，这一级数据传输速度不是最重要的，但是应能传送大容量的信息。

（3）工厂管理层

车间操作员工作站可以通过集线器与车间办公管理网连接，将车间生产数据送到车间管理层。车间管理网作为工厂主网的一个子网，通过交换机、网桥或路由器等连接到厂区骨干网，将车间数据集成到工厂管理层。

S7-300/400 具有很强的通信功能，CPU 模块集成有 MPI 和 DP 通信接口，有 PROFIBUS-DP 和工业以太网的通信模块，以及点对点通信模块。通过 PROFIBUS-DP 或 AS-i 现场总线，CPU 与分布式 I/O 模块之间可以周期性地自动交换数据（过程映像数据交换）。在自动化系统之间，PLC 与计算机和 HMI（人机接口）站之间，均可以交换数据。数据通信可以周期性地自动进行，或基于事件驱动（由用户程序块调用）。

二、S7-300/400 的通信网络

西门子的 PLC 网络是为满足不同控制需要制定的，也为各个网络层次之间提供了互连模块或装置，利用它们可以设计出满足各种应用需求的控制管理网络。其网络系统结构如图 7-14 所示，具体网络类型包括以下几种。

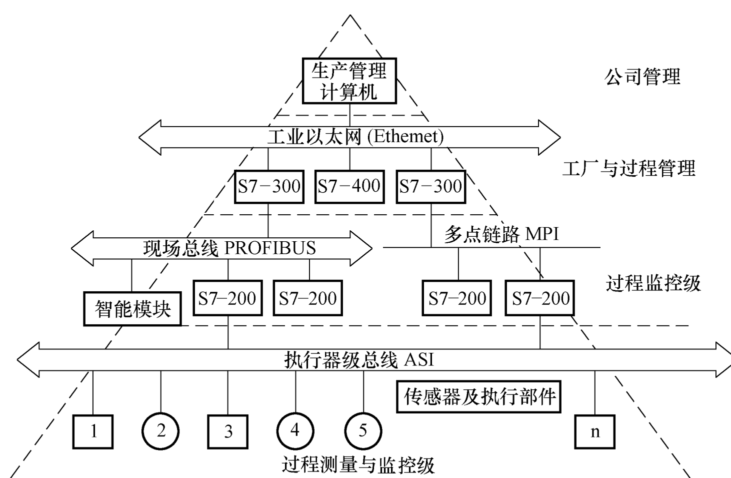


图 7-14　网络系统结构图

1. 工业以太网（Industrial Ethemet）

工业以太网是用于工厂管理层和车间监控层（也称单元层）的通信系统，符合 IEEE 802.3 的国际标准，适用于对时间要求不太严格、需要传送大量数据的通信场合，可以通过网关来连接远程网络。它支持广域的开放型网络模型，可以采用多种传输媒体。西门子的工业以太网的传输速率为 10M/100M bit/s，最多 1024 个网络节点，网络的最大通信范围为 150km。

西门子的 S7 PLC 通过 PROFIBUS 数据链路层协议（工业以太网协议），可以利用它们的通信服务进行数据交换。通信处理器（CP）不会加重 CPU 的通信服务负担，S7-300 最多可以使用 8 个通信处理器，每个通信处理器最多能建立 16 条链路。

2. 通过多点接口（MPI）协议的数据通信

MPI 是多点接口（Multi Point Interface）的简称，S7-300/400 都集成了 MPI 通信协议，MPI 的物理层是 RS-485，最大传输速率为 12Mbit/s，MPI 网采用全局数据（GD）通信模式。PLC 通过 MPI 能同时连接运行 STEP7 的编程器、计算机、人机界面（HMI）及其他 SIMATIC S7、M7 和 C7。接入 MPI 网的设备称为节点，每个 MPI 节点都有自己的 MPI 地址（0～126），同时链接的通信设备数目与 CPU 型号有关，例如：CPU312 为 6 个，CPU418 为 64 个。MPI 接线图如

图 7-15 所示。

3. PROFIBUS

工业现场总线 PROFIBUS 是用于车间级监控和现场层的通信系统，它符合 IEC 61158 标准，具有开放性，符合该标准的各厂商生产的设备都可以接入同一网络中。S7-300/400PLC 可以通过通信处理器或集成在 CPU 上的 PROFIBUS-DP 接口连接到 PROFIBUS-DP 网络上。

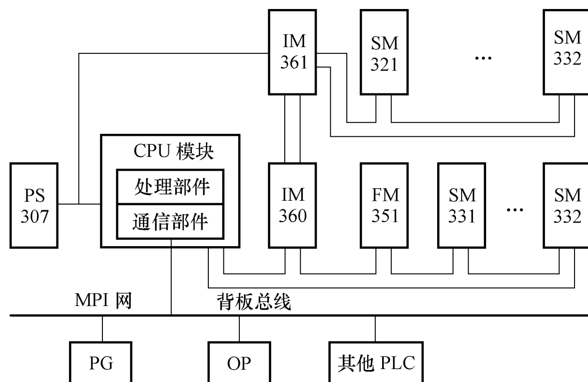


图 7-15 MPI 接线图

如果 PROFIBUS 网络采用 FMS 协议，工业以太网采用 TCP/IP 或 ISO 协议，S7-300 可以与其他公司的设备实现数据交换。带有 PROFIBUS-DP 主站/从站接口的 CPU 能够高速和方便地实现分布式 I/O 的控制。

PROFIBUS 的物理层是 RS-485，最大传输速率为 12Mbit/s，最多可以和 127 个网络节点进行数据交换。网络可以通过中继器来延长网络的通信距离，但是最多只可以串接 10 个中继器。

CP P342/343 是设备连接中的通信处理器，通过它可以将 S7-300 跟 PROFIBUS-DP 或工业以太网相连，常见的连接设备有工业 PC、个人计算机、人机界面（HMI）、S7-300/400 等设备。在连接过程中，设备根据功能的不同，具有主从站之分。主站设备通常包括带有 PROFIBUS-DP 的 S7-300/400 的 CPU、CP 342-5 或 CP343-5、带有 DP 或 DP 通信处理器的 C7、编程器（PG）和操作员界面（OP）；从站设备通常包括带有通信处理器 CP 342-5 的 S7-300、带 DP 接口的 S7-300CPU、带有通信处理器 CP 443-5 的 S7-400 等。

4. 点对点连接

点对点连接（Point-to-Point Connections）可以连接两台 S7 PLC 和 S5 PLC，以及计算机、机器人控制系统、打印机、扫描仪等非西门子设备。使用 CP 340、CP 341 和 CP 441 通信处理模块，或通过 CPU 313C-2PtP 和 CPU 314C-2PtP 集成的通信接口，可以建立起经济而方便的点对点连接。

点对点通信具有 RS-232C 和 RS-485 接口，同时还具备连接 20mA 的 TTY。全双工模式（RS-232C）的最高传输速率为 19.2k bit/s，半双工模式（RS-485）的最高传输速率为 38.4kbit/s。

5. 通过 AS-i 的过程通信

执行器—传感器接口（Actuator-Sensor-Interface，AS-i），是位于自动控制系统最底层的网络，用来连接有 AS-i 接口的现场二进制设备，只能传送少量的数据，例如开关的状态。CP 342-2 通信处理器是用于 S7-300 和分布式 I/O ET 200M 的 As-i 主站，它最多可以连接 62 个数字量或 31 个模拟量 AS-i 从站。通过 AS-i 接口，每个口最多可以访问 248 个数字量输入和 186 个数字量输出。通过内部集成的模拟量处理程序，可以像处理数字量值那样非常容易地处理模拟量值。表 7-2 是以上几种 S7-300/400 通信网络的性能表。

表 7-2 西门子 PLC 通信网络性能表

网络	S1	L2/L2FO	H1/H1FO	H3
标准	SAI 规范 IEC TG 17B	PROFIBUS DINE 19245	以太网 IEEE 802.3	FDDI ISO 9314
访问模式	主机—从机	主从式令牌传递	CSMA/CD	令牌环
传输速率	31 个从机时 5ms	9.6～1500kbit/s	10Mbit/s	100Mbit/s

（续）

网络	S1	L2/L2FO	H1/H1FO	H3
传输介质	无屏蔽双绞电缆	L2：屏蔽双绞电缆 L2FO：玻璃或纤维光缆	H1：双绞电缆 H1FO：玻璃或纤维光缆	玻璃光缆
最大站数 网络尺寸 （大约）	31 个从机(4 通道) 线长 100m	127 个 L2：9.6km L2FO：23.8km	1024 个 H1：1.5km H1FO：4.6km	500 个 100km 环周长
拓扑	线形、树形	线、树、星形	线、树、星形	环形、星形
协议	ASI	SINEC L2-FMS SINEC L2-DP 等	SINEC H1-TF SINEC 等	
应用	执行器—传感器—驱动器	单元网络、现场网络	单元网络、局域网络	主干网络

三、通信的分类

S7 通信可以分为全局数据通信、基本通信及扩展通信 3 类。

1. 全局数据（GD）通信

通信通过 MPI 接口在 CPU 间循环交换数据，用全局数据表来设置各 CPU 之间需要交换的数据存放的地址区和通信的速率，通信是自动实现的，不需要用户编程。当过程映像被刷新时，在循环扫描检测点进行数据交换。S7-400 的全局数据通信可以用 SFC 来起动。全局数据可以是输入、输出、标志位（M）、定时器、计数器和数据区。

S7-300 CPU 每次最多可以交换 4 个包，每个数据包最大为 22B，最多可以有 16 个 CPU 参与数据交换。

S7-400 CPU 可以同时建立最多 64 个站的连接，MPI 网络最多 32 个节点。任意两个 MPI 节点之间可以串联 10 个中继器，以增加通信的距离。每次程序循环最多 64B，最多 16 个数据包。通过全局数据通信，一个 CPU 可以访问另一个 CPU 的数据块、存储器位和过程映像等。全局通信用 STEP 7 中的 GD 表进行组态。对 S7、M7 和 C7 的通信服务可以用系统功能块来建立。

MPI 默认的传输速率为 187.5kbit/s，与 S7-200 通信时只能指定 19.2kbit/s 的传输速率。通过 MPI 接口，CPU 可以自动广播其总线参数组态（例如波特率）。然后 CPU 可以自动检索正确的参数，并连接至一个 MPI 子网。

2. 基本通信（非配置通信）

这种通信可以用于所有 S7-300/400CPU，通过 MPI 或站内的 K 总线（通信总线）来传送最多 76B 的数据。在用户程序中用系统功能（SFC）来传送数据。在调用 SFC 时，通信连接被动态地建立，CPU 需要一个自由的连接。

3. 扩展通信（配置通信）

这种通信可以用于所有的 S7-300/400 CPU，通过 MPI、PROFIBUS 和工业以太网最多可以传送 64KB 的数据。通信是通过系统功能块（SFB）来实现的，支持有应答的通信。在 S7-300 中可以用 SFB 15“PUT”和 SFB 14“GET”来写出或读入远端 CPU 的数据。扩展的通信功能还能执行控制功能，例如控制通信对象的起动和停机。这种通信方式需要用连接表配置连接，被配置的连接在站起动时建立并一直保持。

四、MPI 全局数据通信

西门子有两种硬件 MPI 连接器，一种带有 PG 接口，一种没有 PG 接口。在和通信处理器连接时，在通信处理器上应插一块 MPI 卡或 PC/MPI 适配器，对于终端的站，应将其连接器上的终端电阻开关合上，以接入终端电阻。

通过 MPI 可以访问 PLC 所有智能模块。STEP7 的用户界面提供了 GD 通信组态功能，使得通

信的组态非常简单。在 S7-300PLC 中，MPI 总线与 K 总线（通信总线）连接在一起，S7-300 机架上 K 总线的每一个节点（功能模块 FM 和通信处理器）也是 MPI 的一个节点，有自己的 MPI 地址。在 S7-400 中，MPI（187.5kbit/s）通信模式被转换为内部 K 总线（10.5Mbit/s）。S7-400PLC 只有 CPU 有 MPI 地址，其他智能模块没有独立的 MPI 地址。

通过 GD 通信，一个 CPU 可以访问另一个 CPU 的位存储器、输入输出映像区、定时器、计数器和数据块中的数据。对 S7、M7 和 C7 的通信服务可以用系统功能块来建立。MPI 通信默认的传输速率为 187.5kbit/s 或 1.5Mbit/s，与 S7-200 通信时，只能指定为 19.2kbit/s。两个相邻节点间的最大传送距离为 50m，加中继器后为 1000m，使用光纤和星形连接时为 23.8km。

通过 MPI，CPU 可以自动广播其总线参数组态，然后 CPU 可以自动检索正确的参数，并连接至一个 MPI 子网。每个 MPI 分支网有一个分支网络号，以区别不同的 MPI 分支网。在 MPI 网运行期间，不能插拔模块。

全局数据（GD）通信方式以 MPI 分支网为基础，是为循环地传送少量数据而设计的。GD 通信方式仅限于同一分支网的 S7 系列 PLC 的 CPU 之间，构成的通信网络简单，但只实现两个或多个 CPU 间的数据共享。S7 程序中的功能块（FB）、功能（FC）、组织块（OB）都能用绝对地址或符号地址来访问 GD。在一个 MPI 分支网络中，最多有 16 个 CPU 能通过通信交换数据。在分支网上实现全局数据共享的多个 CPU，至少有一个是数据的发送方，有一个或多个是数据的接收方。发送或接收的数据称为全局数据（GD），全局数据包（GD 包）分别定义在发送方和接收方 CPU 的存储器中，定义在发送方 CPU 中的称为发送 GD 包，定义在接收方 CPU 中的称为接收 GD 包。依靠 GD 包，为发送方和接收方的存储器建立了映射关系。在 PLC 操作系统的作用下，发送 CPU 在它的扫描循环的末尾发送 GD 包，接收 CPU 在它的扫描循环的开头接收 GD 包。这样，发送 GD 包中的数据，对于接收方来说是透明的，接收方对 GD 包的访问，相当于对发送 GD 包的访问。

1. 全局数据包

GD 可以由位、字节、字、双字或相关数组组成，它们是全局数据的元素。全局数据的元素可以定义在 PLC 的位存储器、输入、输出、定时器、计数器和数据块中，例如：I4.2（位）、QB3（字节）、MW20（字）、MD8（双字）等都是一些合法的 GD 元素。具有相同发送者和接收者的全局数据元素可以集合成一个全局数据包。一个全局数据包（GD 块）由一个或几个 GD 元素组成。

2. 全局数据环

所谓全局数据环（GD 环），是指全局数据块的一个确切的分布回路，这个环中的 CPU 既能向环中其他 CPU 发送数据，也能从环中其他 CPU 接收数据。典型的 GD 环有以下两种。

1）由两个 CPU 构成的 GD 环，一个 CPU 既能向另一个 CPU 发送数据块又能接收数据块，类似全双工点对点的通信方式。

2）由两个以上 CPU 组成的 GD 环，一个 CPU 作 GD 块发送方时，其他的 CPU 只能是该 GD 块的接收方，一对多广播通信方式。

同一个 GD 环中的 CPU 可以向环中其他的 CPU 发送数据或接收数据。在一个 MPI 网络中，可以建立多个 GD 环。每个数据包有数据包编号，数据包中的变量有变量号。例如 GD 4.3.3 是 4 号 GD 环、3 号 GD 包中的 3 号数据。

五、MPI 网络的组建

MPI 网络连接如图 7-16 所示，它包括 S7-300/400 系列的 CPU、OP 及 PG 等。MPI 网络的第

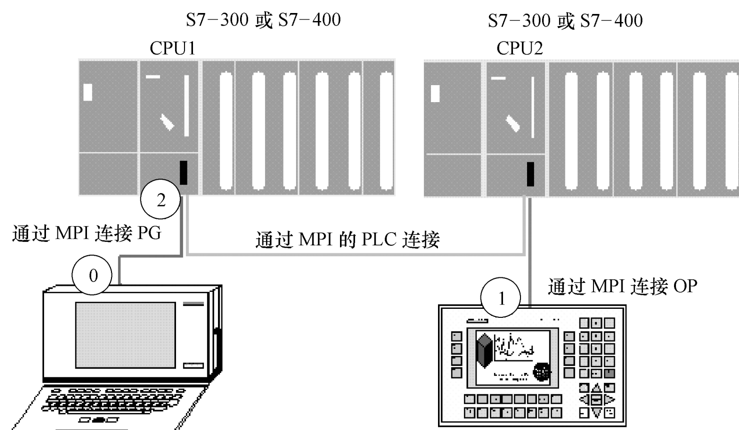


图 7-16　MPI 网络连接

一个及最后一个节点应接入通信终端匹配电阻。如需要添加一个新节点时，应该切断 MPI 网的电源。

连接 MPI 网络时常用到两个网络部件：网络插头和网络中继器。

插头是 MPI 网连接节点的 MPI 接口与网络电缆的连接器。为了保证网络通信质量，网络插头或中继器上都设计了终端匹配电阻。组建通信网络时，在网络拓扑分支的末端节点需要接入浪涌匹配电阻，如图 7-17 所示。

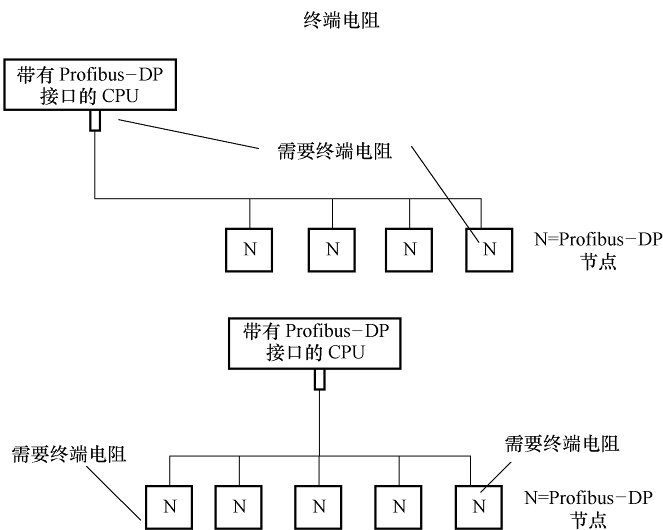


图 7-17　浪涌匹配电阻

对于 MPI 网络，节点间的连接距离是有限制的，从第一个节点到最后一个节点的最长距离仅为 50m。对于一个要求较大区域的信号传输或分散控制的系统，采用两个中继器（或转发器、重复器）可以将两个节点的距离增大到 1000m，但是两个节点之间不应再有其他点。

在采用分支线的结构中，分支线的距离是与分支线的数量有关的，分支线增加，最大距离将缩短。对于 MPI 网络系统，在接地的设备和不接地的设备之间连接时，应该注意 RS-485 的使用，如果 RS-485 中继器所在段中的所有节点都是以接地电位方式运行的，则其是接地的。如果 RS-485 中继器所在段中的所有节点都是以不接地电位方式运行的，则其是不接地的。中继器可以放大信号、扩展节点间的连接距离，也可用于抗干扰隔离，如作不接地的节点与接地 MPI 编程装置的隔离器。要想在接地的结构中运用中继器，就不应该取下 RS-485 中继器上的跨接线。如果需要让其不接地运行，则应该取下跨接线，而且中继器要有一个不接地的电源。在 MPI 网上，如果有一个不接地的节点，那么可以将一台不接地的编程装置接到这个节点上；要想用一个接地的编程装置去操作一个不接地的节点，应该在两者之间接有 RS-485 中继器。

六、MPI 网络组态

1. MPI 网络的组态

1）在 SIMATIC 管理器中生成一个 S7 的项目。

2）自动生成一个 MPI 网络对象。

3）在该项目下至少配置两个可全局通信的模块（例如 S7 的 CPU）。组态 CPU 时，必须定义 CPU 是连接在 MPI 网络上，并分配各自的 MPI 地址。

4）分别向每个 CPU 中下装配置数据。

5）用网络电缆将 CPU 模块连接起来。

6）利用 SIMATIC 管理器的"Accessible Nodes"功能来检查组网是否正确。

硬件配置如图 7-18 所示。

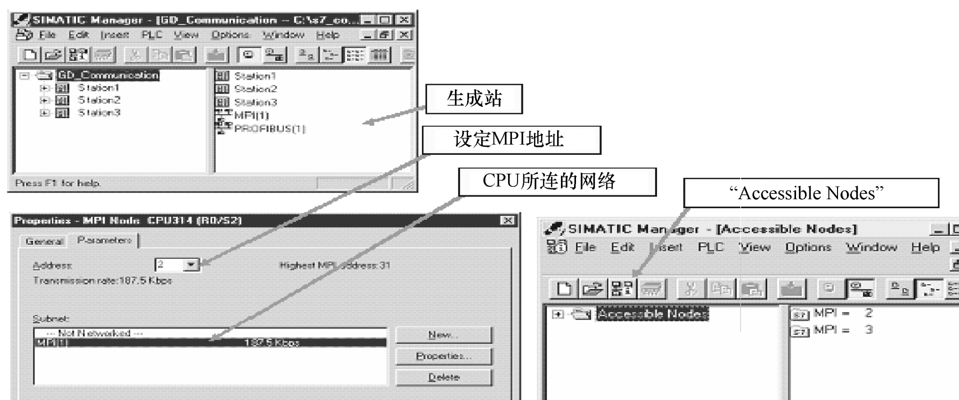


图 7-18　硬件配置

2. 生成和填写 GD 表

GD 通信用全局数据表（GD 表）来设置。在 GD 表中输入要交换数据的 CPU 及数据的地址。还可以定义扫描率（scan rate）和存储状态信息的一个双字。GD 表生成图如图 7-19 所示。

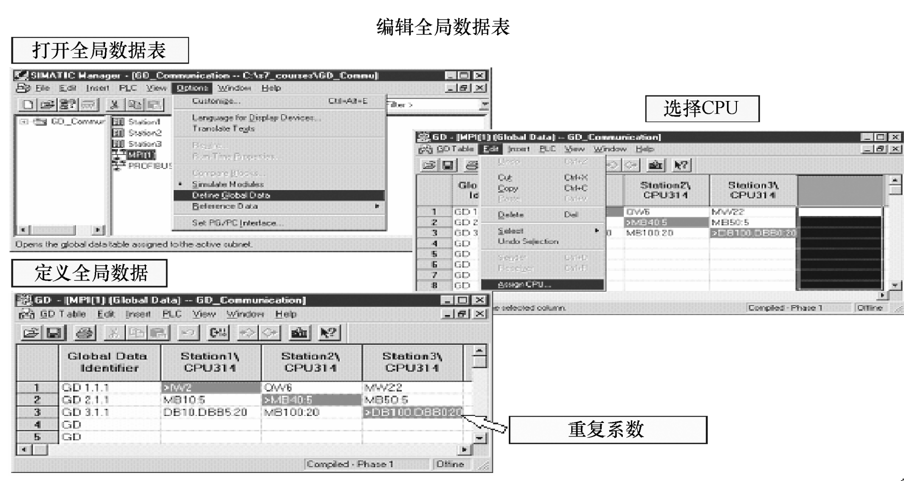


图 7-19　GD 表生成图

生成和填写 GD 表的过程如下：

1）打开项目并选择 MPI 网络对象。

2）选择菜单命令“Options→Define Global Data”，将产生一个新数据表或打开一个已经存在的数据表。

3. 填写 GD 表

在每栏中输入每个 CPU 中用于全局数据交换的地址，步骤如下：

1）首先为表中每栏指定 CPU，方法是用鼠标单击栏首选中它，然后选择菜单命令“Edit→Assign CPU”。

2）在对话框中选择 CPU，然后点“OK”确认。

3）在下面的行中输入要发送的全局数据。用“F2”键可以编辑表中的每一个单元。变量的复制因子用来定义发送数据区的长度。在本例中，定义了从 DB100（Station_3）的 DBB0 开始的 20B。

4）在每行中定义数据的发送方，方法是选择相应的单元，然后在工具条上单击图标“Select as Sender”。

4. 编译 GD 表

现在可以根据 GD 表编译生成配置数据了，配置数据的产生分为两个阶段：

选择菜单功能 GD Table→Compile 起动第一次编译。此时单独的变量被集合成数据包，同时生成数据组。

相应的数据组号码、数据包号码和变量号码将显示在第一栏中：

GD 1. 1. 1：第一数据组中第一数据包里的第一个变量。

GD 1. 2. 1：第一数据组中第二数据包里的第一个变量。

GD m. 3. n：第 n 数据组中第 3 数据包里的第 m 个变量。

第一次编译后，生成了全局数据组和数据包。接着可以为每个数据包定义不同的扫描率（scan rates）以及存储状态信息的地址。然后必须再次编译，使扫描率及状态信息存储地址等包含在配置数据中。

（1）扫描率设置

可以用菜单命令“View→Scan Rates”来选择不同的数值（对于发送方和接收方为从 1 ~ 255，在 S7-400 中选择 0 表示纯事件驱动的发送和接收通信）。

（2）状态设置

如果想通报是否数据已被正确地传送，可以给每个数据包定义一个双字来存储状态信息。方法是选择菜单命令“View→GD Status”。CPU 的操作系统将把检查信息存在该双字中，GD 表编译图如图 7-20 所示。

5. 下装全局通信配置数据

第二次编译完配置数据后，可以将其下装到 CPU 中，如图 7-21 所示。步骤如下：

1）将所有的有关 CPU 切换为 STOP 状态。

2）选择菜单命令“PLC→Download”。

3）成功地传递了配置数据后，将相关 CPU 切换回 RUN 状态。全局数据开始自动地循环交换。

全局数据交换的过程如下：

1）在循环周期的末尾发送方 CPU 发送数据。

2）在循环周期的开始，接收方 CPU 将得到的数据传送到相应的地址区域中。通过定义扫描率可以设置两次发送或接收数据所间隔的循环周期数。

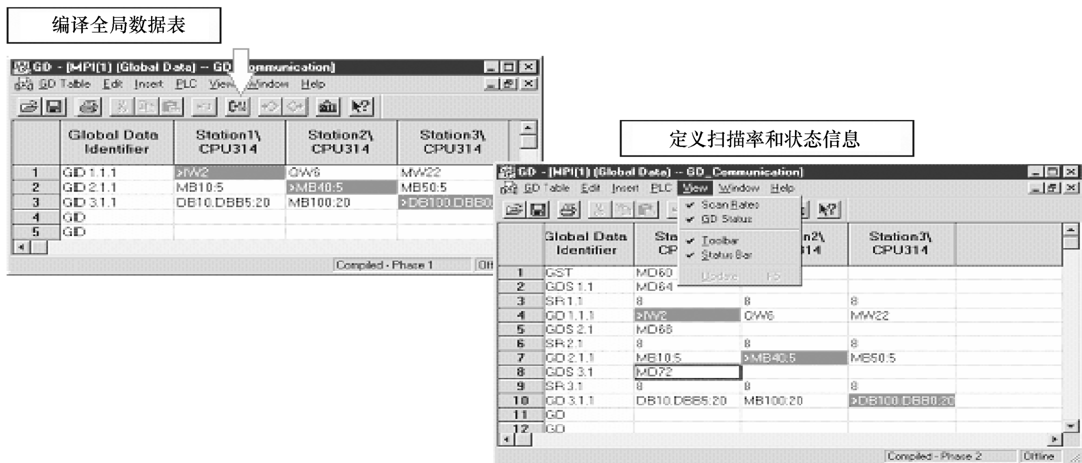


图 7-20　GD 表编译图

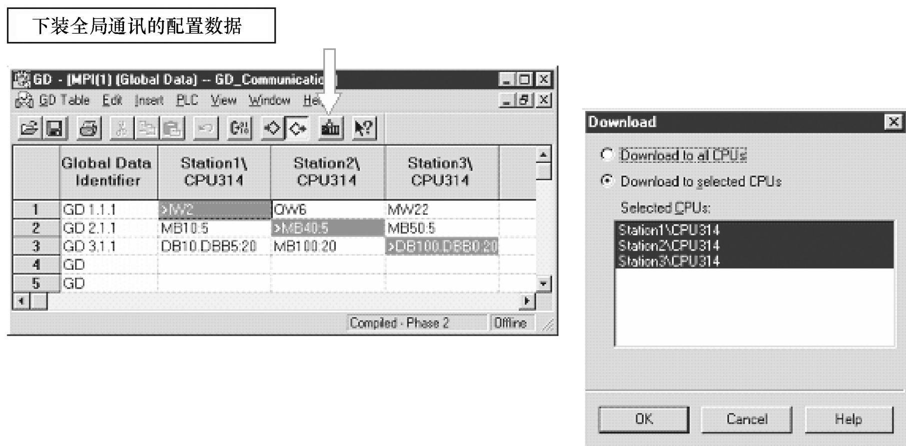


图 7-21　下装全局通信配置数据

6. 全局通信的状态信息

(1) 状态指示

可以为参与全局通信的每个 CPU 的每个数据包定义一个状态双字。在全局数据表中状态双字的标识为“GDS”。

(2) 分析状态双字

如果为状态双字（GDS）分配了 CPU 地址（例如 MD120），就可以在用户程序或 PG 中分析通信的状态。

(3) 状态双字的结构

全局数据的状态双字是由位形式的信息组成的。上图指示了如果位被置位所表达的意义。被置位的位将保持其状态直到被用户程序或 PG 输入所复位。没有标注的位目前未被使用且无意义。状态信息要求存储在一个双字里，所以在本例中使用 MD 120。

(4) 集团信息

STEP 7 为所有的数据组提供了集团状态信息（GST）。集团状态信息也存储在一个双字里，结构与状态信息（GDS）相同，是将所有状态字进行或运算所得的结果。整个全局数据通信状态如图 7-22 所示。

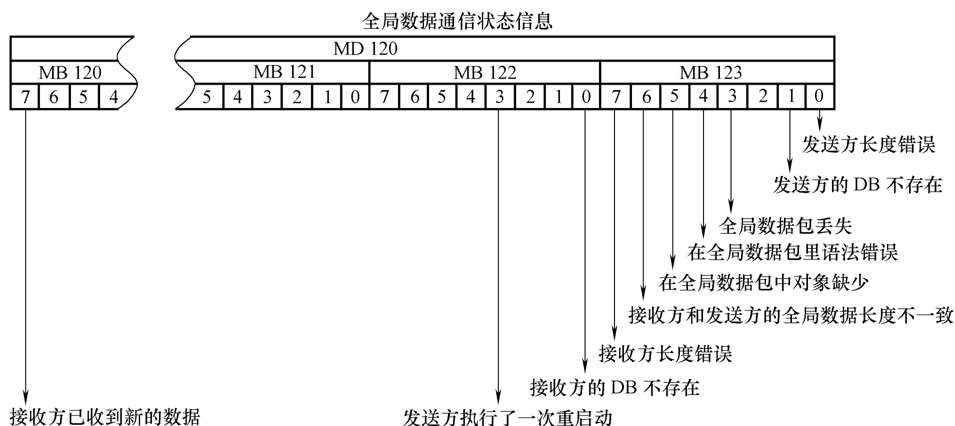


图 7-22 全局数据通信状态

第四节 PROFIBUS 概述

PROFIBUS 已被纳入现场总线的国际标准 IEC 61158 和欧洲标准 EN 50170，并于 2001 年被定为我国的国家标准 JB/T 10308. 3—2001。PROFIBUS 在 1999 年 12 月通过的 IEC 61156 中称为 Type 3，PROFIBUS 的基本部分称为 PROFIBUS-V0。在 2002 年新版的 IEC 61156 中增加了 PROFIBUS-V1、PROFIBUS-V2 和 RS-485IS 等内容。新增的 PROFInet 规范作为 IEC 61158 的 Type10。截止至 2003 年年底，安装的 PROFIBUS 节点设备已突破了 1 千万个，在中国超过 150 万个。

一、PROFIBUS 的组成

1. PROFIBUS-FMS（Fieldbus Message Specification，现场总线报文规范）

主要用于系统级和车间级的不同供应商的自动化系统之间传输数据，处理单元级（PLC 和 PC）的多主站数据通信。

2. PROFIBUS-DP（Decentralized Periphery，分布式外部设备）

PROFIBUS-DP 用于自动化系统中单元级控制设备与分布式 I/O（例如 ET 200）的通信。主站之间的通信为令牌方式，主站与从站之间为主从方式，以及这两种方式的混合。

3. PROFIBUS-PA（Process Automation，过程自动化）

用于过程自动化的现场传感器和执行器的低速数据传输，使用扩展的 PROFIBUS-DP 协议。传输技术采用 IEC 1158-2 标准，可以用于防爆区域的传感器和执行器与中央控制系统的通信。使用屏蔽双绞线电缆，由总线提供电源。

此外基于 PROFIBUS，还推出了用于运动控制的总线驱动技术 PROFI-drive 和故障安全通信技术 PROFI-safe。

二、PROFIBUS 的物理层

可以使用多种通信介质（电、光、红外、导轨以及混合方式）。传输速率为 9.6kbit/s ~ 12Mbit/s，假设 DP 有 32 个站点，所有站点传送 512bit/s 输入和 512bit/s 输出信息，在 12-Mbit/s 时只需 1ms。每个 DP 从站的输入数据和输出数据最大为 244B。使用屏蔽双绞线电缆时最长通信距离为 9.6km，使用光缆时最长为 90km，最多可以接 127 个从站。可以使用灵活的拓扑结构，支持线形、树形、环形结构以及冗余的通信模型。

1. DP/FMS 的 RS-485 传输

DP 和 FMS 使用相同的传输技术和统一的总线存取协议，可以在同一根电缆上同时运行。DP/FMS 符合 EIA RS-485 标准（也称为 H2），采用屏蔽或非屏蔽双绞线电缆，9.6kbit/s ~

12Mbit/s。一个总线段最多 32 个站，带中继器最多 127 个站。A 型电缆，3～12Mbit/s 时为 100m，9.6～93.75kbit/s 时为 1200m。

2. D 型总线连接器

PROFIBUS 标准推荐总线站与总线的相互连接使用 9 针 D 型连接器。A、B 线上的波形相反。信号为 1 时 B 线为高电平，A 线为低电平。

3. 总线终端器

总线终端器的连接如图 7-23 所示。

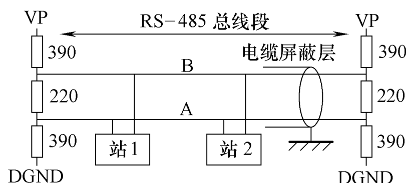


图 7-23 总线终端器的连接

4. DP/FMS 的光纤电缆传输

单芯玻璃光纤的最大连接距离为 15km，价格低廉的塑料光纤为 80m。光链路模块（OLM）用来实现单光纤环和冗余的双光纤环。

5. PA 的 IEC 1158-2 传输

采用符合 IEC 1158-2 标准的传输技术，确保本质安全，并通过总线直接给现场设备供电。数据传输使用曼彻斯特编码线协议（也称 H1 编码）。从 0(－9mA)～1(＋9mA) 的上升沿发送二进制数“0”，从 1～0 的下降沿发送二进制数“1”。传输速率为 31.25kbit/s。传输介质为屏蔽或非屏蔽的双绞线。总线段的两端用一个无源的 RC 线路终端器来终止（100Ω 电阻与 1μF 电容的串联电路），一个 PA 总线段最多 32 个站，总数最多为 126 个。

三、PROFIBUS-DP 设备的分类

1. 1 类 DP 主站

1 类 DP 主站（DPM1）是系统的中央控制器，DPM1 与 DP 从站循环地交换信息，并对总线通信进行控制和管理 PLC、PC 等。

2. 2 类 DP 主站

DP 网络中的编程、诊断和管理设备。DPM2 除了具有 1 类主站的功能外，还可以读取 DP 从站的 I/O 数据和当前的组态数据，可以给 DP 从站分配新的总线地址。

3. DP 从站

1）分布式 I/O（非智能型 I/O）由主站统一编址。

2）PLC 智能 DP 从站（I 从站）：PLC（智能型 I/O）作从站。存储器中有一片特定区域作为与主站通信的共享数据区。

3）具有 PROFIBUS-DP 接口的其他现场设备

四、PROFIBUS 的通信协议

1. PROFIBUS 的数据链路层

PROFIBUS 的总线存取方式如图 7-24 所示。

在总线的存取中，有两个基本要求：

1）保证在确切的时间间隔中，任何一个站点有足够的时间来完成通信任务。

2）尽可能简单快速地完成数据的实时传输，因通信协议增加的数据传输时间应尽量少。

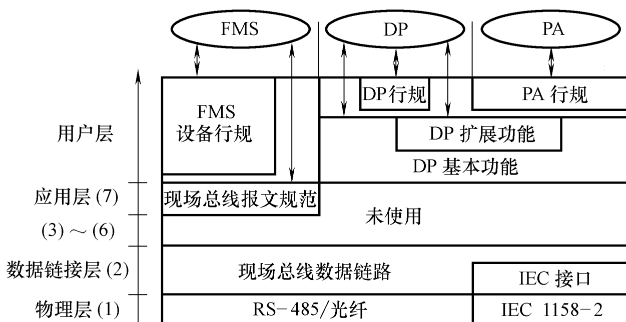


图 7-24 PROFIBUS 的总线存取方式

PROFIBUS 采用主站（Master）之间的令牌（Token）传递方式和主站与从站（Slave）之间的主-从方式，如图

7-25 所示。

当某主站得到令牌报文后可以与所有主站和从站通信。在总线初始化和起动阶段建立令牌环。在总线运行期间，从令牌环中去掉有故障的主动节点，将新上电的主动节点加入到令牌环中。监视传输介质和收发器是否有故障，检查站点地址是否出错，以及令牌是否丢失或有多个令牌。

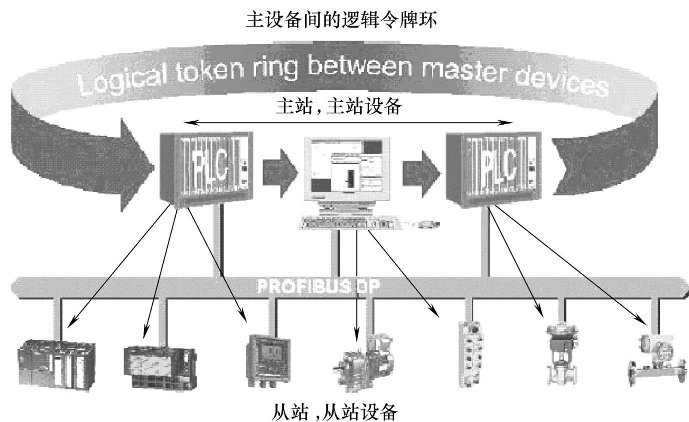


图 7-25　数据传输方式

DP 主站与 DP 从站间的通信基于主-从原理，DP 主站按轮询表依次访问 DP 从站。报文循环由 DP 主站发出的请求帧（轮询报文）和由 DP 从站返回的响应帧组成。

2. PROFIBUS-DP

DP 的功能共有 3 个，即 DP-V0、DP-V1 和 DP-V2。

（1）基本功能（DP-V0）

① 总线存取方法

② 3 级诊断功能

③ 保护功能

只有授权的主站才能直接访问从站。主站用监控定时器监视与从站的通信。从站用监控定时器检测与主站的数据传输。

④ 通过网络的组态功能与控制功能

动态激活或关闭 DP 从站，对主站进行配置，设置从站的地址、I/O 数据的格式、诊断报文的格式，检查 DP 从站的组态。

⑤ 同步与锁定功能

⑥ DPM1 和 DP 从站之间的循环数据传输

⑦ DPM1 和系统组态设备间的循环数据传输

（2）DP-V1 的扩展功能

① 非循环数据交换

主站与从站之间的非循环数据交换功能，可以用来进行参数设置、诊断和报警处理。

② 基于 IEC 61131-3 的软件功能块

③ 故障-安全通信（PROFIsafe）

PROFIsave 定义了与故障-安全有关的自动化任务，以及在 PROFIBUS 上的通信。考虑了数据的延迟、丢失、重复，不正确的时序、地址和数据的损坏。

补救措施：输入报文帧的超时及其确认；发送者与接收者之间的口令；CRC 校验。

④ 扩展的诊断功能

DP 从站通过诊断报文将报警信息传送给主站，主站收到后发送确认报文给从站。从站收到后只能发送新的报警信息。

（3）DP-V2 的扩展功能

① 从站与从站之间的通信

广播式数据交换实现了从站之间的通信，从站作为出版者（Publisher），不经过主站直接将

信息发送给作为订户（Subscribers）的从站。

② 同步（Isochronous）模式功能

主站与从站之间的同步，误差小于 1ms。所有设备被周期性地同步到总线主站的循环。

③ 时钟控制与时间标记（Time Stamps）

主站将时间发送给所有的从站，误差小于 1ms。

④ HARTonDP

⑤ 上载与下载（区域装载）

用少量的命令装载任意现场设备中任意大小的数据区。

⑥ 功能请求（Function Invocation）

用于 DP 从站的起动、停止、返回、重新起动和功能调用。

⑦ 从站冗余

冗余的从站有两个 PROFIBUS 接口。在主要从站出现故障时，后备从站接管它的功能。

3. PROFINet

PROFINet 以互联网和以太网标准为基础，建立了 PROFIBUS 与外部系统的透明通道。PROFINet 首次明确了 PROFIBUS 和工业以太网之间数据交换的格式，使跨厂商、跨平台的系统通信问题得到了彻底的解决。PROFINet 提供了一种全新的工程方法，即基于组件对象模型（COM）的分布式自动化技术；以微软的 OLE/COM/DCOM 为技术核心，最大程度地实现了开放性和可扩展性，向下兼容传统工控系统，使分散的智能设备组成的自动化系统模块化。PROFINet 指定了 PROFIBUS 与国际 IT 标准之间的开放和透明的通信；提供了包括设备层和系统层的完整系统模型，保证了 PROFIBUS 和 PROFINet 之间的透明通信。

PROFINet 的通信机制

在 PROFINet 中，每个设备都被看作是一个具有组件对象模型（Component Object Model，COM）接口的自动化设备，系统通过调用 COM 接口来实现设备功能。组件模型使不同厂家的设备具有良好的互换性和互操作性。COM 对象之间通过 DCOM（分布式 COM）连接协议进行互联和通信。传统的 PROFIBUS 设备通过代理设备（Proxy）与 PROFINet 中的 COM 对象进行通信。COM 对象之间的调用是通过对象链接与嵌入（Object Linking and Embedding，OLE）自动化接口实现的。组件技术为企业管理人员通过公用数据网络访问过程数据提供了方便，PROFINet 使用了 IT 技术，支持从办公室到工业现场的信息集成，其通信连接图如图 7-26 所示。

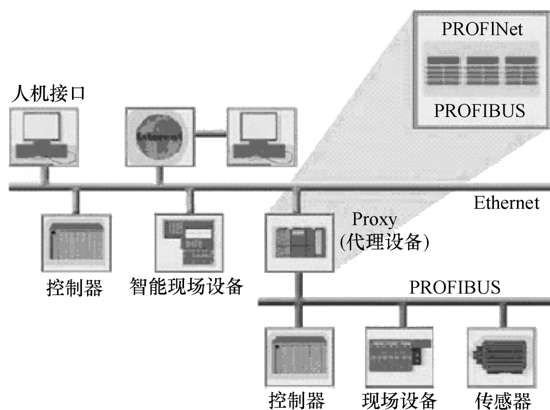


图 7-26 PROFINet 通信连接图

五、基于组态的 PROFIBUS 通信

1. PROFIBUS-DP 从站的分类

1）紧凑型 DP 从站：ET 200B 模块系列。

2）模块式 DP 从站：ET 200M，可以扩展 8 个模块。在组态时 STEP 7 自动分配紧凑型 DP 从站和模块式 DP 从站的 I/O 地址。

3）智能从站（I 从站）：某些型号的 CPU 可以作 DP 从站。智能 DP 从站提供给 DP 主站的

I/O区域不是实际的 I/O 模块使用的 I/O 区域，而是从站 CPU 专门用于通信的 I/O 映像区。

2. PROFIBUS-DP 网络的组态

主站是 CPU 416-2DP，将 DP 从站 ET 200B-16 DI/16DO、ET 200M 和作为智能从站的 CPU 315-2DP 连接起来，传输速率为 1.5Mbit/s。

（1）生成一个 STEP 7 项目 SIMATIC 管理如图 7-27 所示。

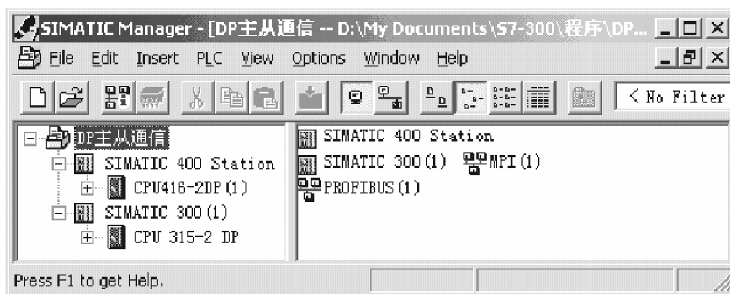


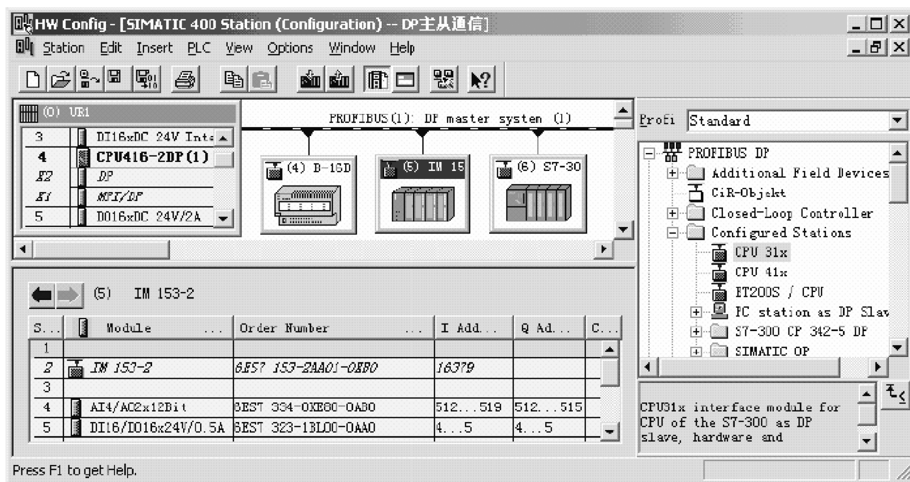
图 7-27　SIMATIC 管理

（2）设置 PROFIBUS 网络

右键点击“项目”对象，生成网络对象 PROFIBUS（1），在自动打开的网络组态工具 NetPro 中，双击图中的 PROFIBUS 网络线，设置传输速率为 1.5Mbit/s，总线行规为 DP。最高站地址使用默认值 126。

（3）设置主站的通信属性

选择 400 站对象，打开 HW Config 工具。双击机架中“DP”所在的行，在“Operating Mode”标签页选择该站为 DP 主站。默认的站地址为 2，如图 7-28 所示。



S...	Module	Order Number	I Add...	Q Ad...	C...
1					
2	IM 153-2	6ES7 153-2AA01-0XB0	16379		
3					
4	AI4/AO2x12Bit	6ES7 334-0XE80-0AB0	512...519	512...515	
5	DI16/DO16x24V/0.5A	6ES7 323-1BL00-0AA0	4...5	4...5	

图 7-28　PROFIBUS 网络组态

（4）组态 DP 从站 ET 200B

组态第一个从站 ET 200B -16DI/16DO。设置站地址为 4，各站的 I/O 自动统一编址。选择监控定时器功能。

（5）组态 DP 从站 ET 200 M

将接口模块 IM 153-2 拖到 PROFIBUS 网络线上，设置站地址为 5。打开硬件目录中的 IM 153-2 文件夹，插入 I/O 模块。

（6）组态一个带 DP 接口的智能 DP 从站

在项目中建立 S7-300 站对象，CPU 315-2DP 模块插入槽 2。默认的 PROFIBUS 地址为 6。设置为 DP 从站。在“HW Config”中保存对 S7-300 站的组态。

（7）将智能 DP 从站连接到 DP 主站系统中

返回到组态 S7-400 站硬件的界面。打开\PROFIBUS-DP\ConfiguredStations（已经组态的站）文件夹，将“CPU 31x”拖到屏幕左上方的 PROFIBUS 网络线上。自动分配的站地址为 6。在“Connection”标签页选中 CPU 315-2DP，点击“Connect”按钮，该站被连接到 DP 网络中。

3. 主站与智能从站主从通信方式的组态

DP 主站直接访问“标准”的 DP 从站（例如 ET 200B 和 ET 200M）的分布式 I/O 地址区。用于主站和从站之间交换数据的 I/O 区不能占据 I/O 模块的物理地址区。

点击 DP 从站对话框中的“Configuration”标签，为主-从通信的智能从站配置 I/O 区地址（见图 7-29）。点击图中的“New”按钮，出现如图 7-30 所示的设置 DP 从站 I/O 区地址的对话框。

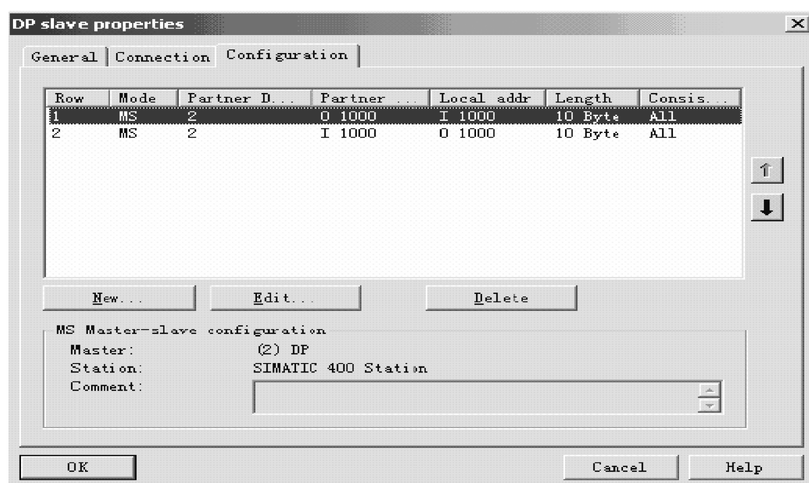


图 7-29 DP 主从通信地址的组态

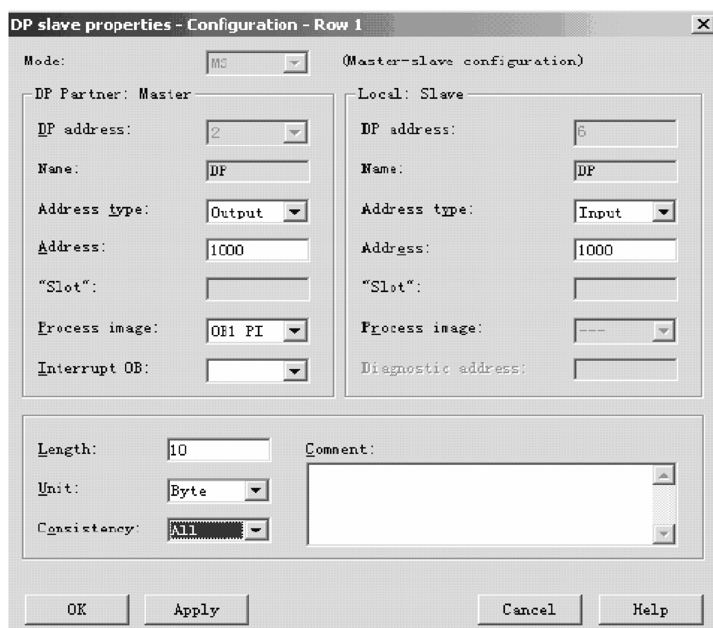


图 7-30 DP 从站属性组态

组态结束后，其状态如图 7-31 所示。

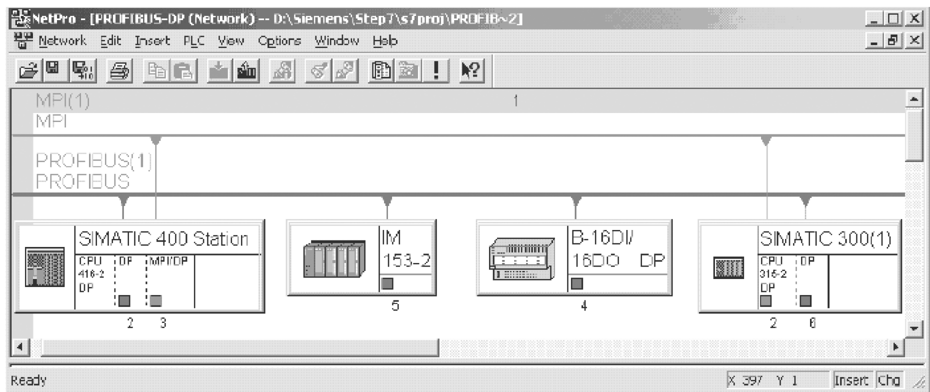


图 7-31　组态后的网络

4. 直接数据交换通信方式的组态

1）单主站系统中 DP 从站发送数据到智能从站（I 从站），如图 7-32 所示。

2）多主站系统中从站发送数据到其他主站（见图 7-33）。

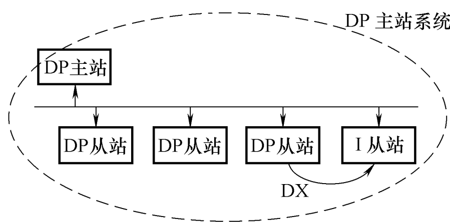


图 7-32　单主站系统

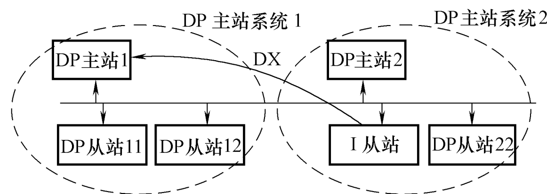


图 7-33　多主站系统 1

3）多主站系统中从站发送数据到智能从站（见图 7-34）。

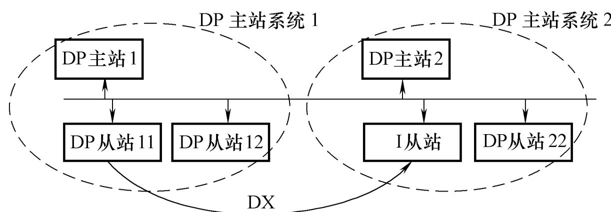


图 7-34　多主站系统 2

第五节　执行器传感器接口网络

AS-i 是执行器-传感器接口（Actuator Sensor Interface，AS-i）的英文缩写，符合 EN 50295 标准，是一种开放标准，世界上领先的执行器和传感器制造商都支持 AS-i。它用于现场自动化设备（即传感器和执行器）的双向数据通信网络，位于工厂自动化网络的最底层。AS-i 特别适用于连接需要传送开关量的传感器和执行器，例如读取各种接近开关、光电开关、压力开关、温度开关、物料位置开关的状态，控制各种阀门、声光报警器、继电器和接触器等，AS-i 也可以传送模拟量数据。

AS-i 网络的数据和辅助电源都经过一根公用电缆传输，借助于一种专门开发的绝缘移动接线法，可在任何位置分接 AS-i 电缆。可利用防护等级为 IP20 和 IP25 的链路，AS-i 直接连接到 PROFIBUS-DP；应用 DP/AS-i 链路，可将 AS-i 用作 PROFIBUS-DP 的一种子网。AS-i 属于单主从式网络，如图7-35所示，每个网段只能有一个主站。

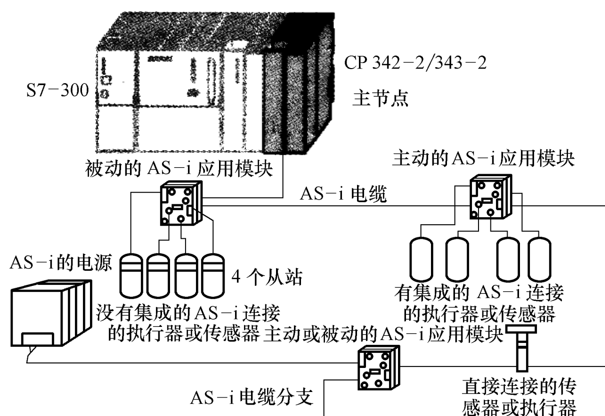


图 7-35 AS-i 网络示意图

主站是网络通信的中心，负责网络的初始化、设置从站的地址和参数等，具有错误校验功能（如发现传输错误将重发报文）。AS-i 从站是 AS-i 系统的输入通道和输出通道，它们仅在被 AS-i 主站访问时才被激活。接到命令时，它们触发动作或者将现场信息传送给主站。AS-i 所有分支电路的最大总长度为 100m，可以加中继器延长。传输介质可以是屏蔽的或非屏蔽的两芯电缆，网络的拓扑结构可为总线、星形或树形。

一、AS-i 的寻址模式

AS-i 的节点（从站）地址为 5 位二进制数，每一个标准从站占一个 AS-i 地址，最多可以连接 31 个从站，地址 0 仅供产品出厂时使用，在网络中应改用别的地址。每一个标准 AS-i 从站可以接收 4 位数据或发送 4 位数据，所以一个 AS-i 总线网段最多可以连接 124 个二进制输入点和 124 个输出点，对 31 个标准从站的典型轮询时间为 5ms，因此 AS-i 适用于工业过程开关量高速输入、输出的场合。

用于 S7-200 的通信处理器 CP 242-2 和用于 S7-300、E 1200M 的通信处理器 CP342-2 属于标准 AS-i 主站。

在扩展的寻址模式中，两个从站分别作为 A 从站和 B 从站，使用相同的地址，这样使可寻址的从站的最大个数增加到 62 个。由于地址的扩展，使用扩展的寻址模式的每个从站的二进制输出减少到 3 个，每个从站最多 4 点输入和 3 点输出。一个扩展的 AS-i 主站可以操作 186 个输出点和 248 个输入点。使用扩展的寻址模式时对从站的最大轮询时间为 10ms。

用于 S7-200 的通信处理器 CP 243-2 和用于 S7-300、ET 200M 的通信处理器 CP 343-2 属于扩展的 AS-i 主站。

二、AS-i 网络接口部件

AS-i 技术的一个显著特征，是用一根公用的双线电缆来传输数据并将辅助电源分配到传感器/执行器。AS-i 网络系统的基本组成部件有：

1）用于如 SIMATIC 5S 和 SIMATIC S7，分布式 I/O ET200U/M/X 或 PC/PG 中央控制单元的主接口。

2）AS-i 异形电缆，AS-i 异形电缆提供机械编码，从而防止极性反置，采用贯穿端子使接触更简便。

3）中继器/扩展器等网络部件。

4）对从站供电的供电单元，AS-i 的电源模块的额定电压为 DC 24V，最大电流为 2A。

5）连接标准传感器/执行器的模块。

6）用于设定从站地址的编程器。

三、AS-i 主站模块

1. CP 243-2

CP 243-2 是 S7-200 CPU 22x 的 AS-i 主站。通过连接 AS-i 可以显著地增加 S7-200 的数字量输入和输出点数，每个 CP 的 AS-i 上最多可以连接 124 个开关量输入和 124 个开关量输出。S7-200 同时可以处理最多2 个 CP 243-2。CP 243-2 与 S7-200 的连接方法与其他扩展模块相同。它有 2 个端子直接连接 AS-i 接口电缆。

前面板的 LED 用来显示模块的状态、所有连接的从站模块的状态以及监控通信电压等。两个按钮用来切换运行状态。在 S7-200 的映像区中，CP 243-2 占用 1 个数字量输入字节作为状态字节，1 个数字量输出字节作为控制字节。8 个模拟量输入字和 8 个模拟量输出字用于存放 AS-i 模拟量 I/O 数据、AS-i 的诊断信息、AS-i 命令与响应数据等。

用户程序用状态字节和控制字节设置 CP 243-2 的工作模式。根据工作模式的不同，CP 243-2 在 S7-200 模拟地址区既可以存储 AS-i 从站的 I/O 数据或诊断值，也可以使用主站调用。CP 243-2 支持扩展 AS-i 特性的所有特殊功能。通过双重地址（A-B）赋值，最多可以处理62 个 AS-i 从站。由于集成了模拟量值处理系统，CP243-2 也可以访问模拟量。

2. CP 343-2

CP 343-2 通信处理器是用于 S7-300 PLC 和分布式功能，最多连接 62 个数字量或 31 个模拟量 AS-i 从站。支持所有 AS-i 主站功能，在前面板上用 LED 显示从站的运行状态、运行准备信息和错误信息，例如 AS-i 电压错误和组态错误。通过 AS-i 接口，每个 CP 最多可以访问 248 个数字量输入和 186 个数字量输出，可以对模拟量值进行处理。CP 342-2 占用 PLC 模拟区的 16 个输入字节和 16 个输出字节。通过它们来读写从站的输入数据和设置从站的输出数据。

3. CP 142-2

AS-i 主站 CP 142-2 用于 ET 200X 分布式 I/O 系统，CP 142-2 通信处理器通过连接器与 ET 200X 模块相连，并使用其标准 I/O 范围。CP142-2 通信处理器通过连接器与 AS-i 网络无需组态，最多 31 个从站可以由 CP142-2（最多 124 点输入和 124 点输出）寻址。

4. DP/AS-i 接口网关模块

DP/AS-i 网关用来连接 PROFIBUS-DP 和 AS-i 网络，DP/AS-interface Link、DP 20 和 AS-interface Link 20E 可以作 DP/AS-i 的网关，后者具有扩展的 AS-i 功能。

5. SMATIC C7 621 AS-i

SIMATIC C7 621 AS-i 把 AS-i 主站 CP 342-2、S7-300 的 CPU 以及 OP3 操作面板组合在一个外壳内，适合于高速方便地执行自动化任务，自带人机界面。这种紧凑型控制器可以直接访问和控制 31 个从站的 124 点数字量输入和 124 点数字量输出，无需在控制器内集成输入和输出，减小了控制器的体积。

6. 用于个人计算机的 AS-i 通信卡 CP 2413

CP 2413 是用于 PC（个人计算机）的标准 AS-i 主站，一台 PC 可以安装 4 块 CP 2413。因为在 PC 中还可以运行以太网和 PROFUBUS 总线接口卡，AS-i 从站提供的数据也可以被其他网络中其他的站使用。SCOPE 是在 PC 中运行的 AS-i 的诊断软件。它可以记录和评估在安装和运行过程中 AS-i 网络中的数据交换。

四、从站模块

AS-i 从站由专用的 AS-i 通信芯片和传感器或执行器部分组成。其中，AS-i 通信芯片内包括 4 个可组态的 I/O 以及 4 个参数输出，AS-i 连接器可以直接集成在执行器和传感器中。4 位 I/O

组态（FO 组态）用来指定从站的哪根数据线用来作为输入、输出或双向传输，从站的类型用标识码来描述。使用 AS-i 从站的参数输出 AS-i 主站可以传送参数值，它们用于控制和切换传感器或执行器的内部操作模式，例如在不同的运行阶段修改标度值。AS-i 从站包括以下功能单元：微处理器、电源供给单元、通信的发送器和接收器、数据 I/O 单元、参数输出单元和 E^2PROM 存储器芯片。

微处理器是实现通信功能的核心，它接收来自主节点的呼叫发送报文，对报文进行解码和出错检查，实现主、从站之间的双向通信，把接收到的数据传送给传感器和执行器，向主站发送响应报文。E^2PROM 存储器用于存储运行参数、指定 I/O 的组态数据、标识码和从站地址等。从站可以带电插拔，短路及过载状态不会影响其他站点的正常通信。

1. AS-i 从站模块

AS-i 从站模块最多可以连接 4 个传统的传感器/执行器。带有集成 AS-i 连接的传感器和执行器可以直接连接到 AS-i 上。Slimljnk 模块的防护等级为 IP 20，可以像其他低压设备一样安装在 DIN 导轨上，或用螺钉固定在控制柜的背板上。IP 65/67 防护等级的 AS-i 从站模块可以直接安装在环境恶劣的工业现场。

2. “LOGO!”微型控制器

“LOGO!”微型控制器是一种微型 PLC，它具有数字量或模拟量输入和输出、逻辑处理器和实时时钟功能。通过内置的 AS-i 模块，“LOGO!”微型控制器可以作为 AS-i 网络中的智能型从站使用，它是 AS-i 网络中有分布式控制器功能的从站。“LOGO!”微型控制器适合于简单的分布式自动化任务（例如门控系统），又可以通过 AS-i 网络将它纳入高端自动化系统。在高端控制系统出现故障时，可以继续进行控制。

3. 紧凑型 AS-i 模块

紧凑型 AS-i 模块是一种具有较高保护等级的新一代 AS-i 模块，包括数字、模拟、气动和 DC24V 电动机起动器模块。模块具有两种尺寸，其保护等级为 IPV6。通过一个集成的编址插孔可以对已经安装的模块编址，所有的模块都可以通过与 S7 系列 PLC 的通信实现参数设置。西门子还提供了模拟量模块，每个模拟量模块有两个通道，分为电流型、电压型、热电阻型传感器输入模块和电流型、电压型执行器输出模块。

4. 气动控制模块

西门子提供两种类型的 AS-i 气动模块，即带两个集成的 3/2 路阀门的气动用户模块和带两个集成的 4/2 路阀门的气动紧凑型模块。模块有单稳和双稳两种类型，集成了作为气动单元执行器的阀门，接收来自气缸的位置信号。

5. 电动机起动器

西门子有 3 种类型的异步电动机起动器，在 AS-i 中作标准从站。防护等级均为 IP 65，有非熔断器保护，可进行可逆起动，起动的异步电动机的最大功率为 4 ~ 5. 5kW。

6. 能源与通信现场安装系统

能源与通信现场安装系统（ECOFAST）是一个开放的控制柜系统解决方案。所有的自动化和相应的安装器件应用标准和接口将数据和动力的传输有机地连成一体。与 AS-i 有关的下列元器件可以集成到 ECOFAST 中：所有的 I/O 模块、安装在电动机接线盒上或电动机附近的可逆起动器和软起动器、集成在电动机上的微型起动器、动力和控制装置（动力电源）、PLC 和 AS-i 主站的组合装置。

7. DC 24V 电动机起动器

K60 AS-i 24V 电动机起动器可以驱动 70W 功率的电动机，它将 DC 24V 起动器及其传感器直

接连接到 AS-i 上。有的有制动器和可选的急停功能。

8. 接近开关

BERO 接近开关可以直接连接到 AS-i 或接口模块上。特殊的感应式、光学和声纳 BERO 接近开关适合直接连接到 AS-i 上。它们集成有 AS-i 芯片，除了开关量输出之外，还提供其他信息，例如开关范围和线圈故障。通过 AS-i 电缆可以对这些智能 BERO 设置参数。

9. 按钮和 LED

SIGNUM 3SB4 是一个具有 AS-i 接口的完整的操作员通信系统人机界面。带灯的指令按钮通过 AS-i 电缆供电，通过特殊的 AS-i 从站和独立的辅助电源，可以实现控制设备的单个连接，每个设备最多可以连接 28 个常开触点和 7 个信号输出点。

五、AS-i 的主从通信方式

AS-i 是单主站系统，AS-i 通信处理器（CP）作为主站控制现场的通信过程，主站一个接一个地轮流询问每一个从站，询问后等待从站的响应。

地址是 AS-i 从站的标识符。可以用专用的定址（Addressing）单元或主站来设置各从站的地址。

AS-i 使用电流调制的传输技术保证了通信的高可靠性。主站如果检测到传输错误或从站的故障，将会发送报文给 PLC，提醒用户进行处理。在正常运行时增加或减少从站向其他从站的通信。

扩展的 AS-i 接口技术规范 V2.1 最多允许连接 62 个从站，主站可以对模拟量进行处理。AS-i的报文主要有主站呼叫发送报文和从站应答（响应）报文，主站的请求由 14 个数据位组成，如图 7-36所示。

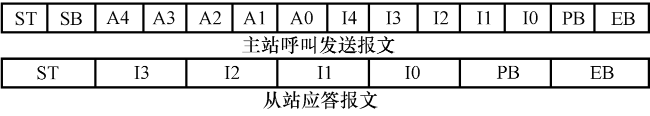


图 7-36　通信报文

在主站呼叫发送报文中，ST 是起始位，其值为 0。SB 是控制位，为 0 或为 1 时分别表示传送的是数据或命令。A4 ~ A0 是从站地址，I4 ~ I0 为数据位。PB 是奇偶校验位，在报文中不包括结束位在内的各位中 1 的个数应为偶数。EB 是结束位，其值为 1。在 7 个数据位组成的从站应答报文中，ST、PB 和 EB 的意义与取值与主站呼叫发送报文的相同。

主站通过呼叫发送报文，可以完成下列功能：

1）数据交换：主站通过报文把控制指令或数据发送给从站，或让从站把测量数据上传给主站。

2）设置从站的参数：例如设置传感器的测量范围，激活定时器和改变测量方法等。

3）删除从站地址：把被呼叫的从站地址暂时改为 0。

4）地址分配：只能对地址为 0 的从站分配地址。从站把新地址存放在 EEPROM 中。

5）复位功能：把被呼叫的从站恢复为初始状态时的地址。

6）读从站的 I/O 配置。

7）读取从站的 ID（标识符）代码。

8）状态读取：读取从站的 4 个状态位，以获得在寻址和复位时出现的错误的信息。

9）状态删除：读取从站的状态并删除其内容。

六、AS-i 的工作模式

1. 初始化

初始化为 AS-i 的离线阶段。模块上电后或被重新起动后被初始化，在此阶段设置主站的基

本状态，而所有从站的输入和输出数据的映像被设置为 0（未激活）。上电后，组态数据被复制到参数区，后面的激活操作可以使用预置的参数。如果主站在运行中被重新初始化，参数区中可能已经变化的值被保持。

2. 起动

在起动阶段，主站检测 AS-i 电缆上连接有哪些从站以及它们的型号。厂家制造 AS-i 从站时通过组态数据，将从站的型号永久性地保存在从站中，主站可以请求上传这些数据。组态文件中包含了 AS-i 从站的 I/O 分配情况和从站的类型（ID 代码），主站将检测到的从站信息存放在从站表中。

3. 激活

在激活阶段，主站检测到 AS-i 从站后，通过发送特殊的呼叫，激活这些从站。主站处于组态模式时，所有地址不为 0 的，被检测到的从站被激活。在这一模式下，可以读取实际的值并将它们作为组态数据保存。

主站处于保护模式时，只有储存在主站的组态中的从站被激活。如果在网络上发现的实际组态不同于期望的组态，主站将显示出来。主站把激活的从站存入被激活的从站表中。

4. 运行

起动阶段结束后，AS-i 主站切换到正常循环的运行模式。

1）数据交换在正常模式下，主站将周期性地发送输出数据给各从站，并接收它们返回的应答报文，即输入数据。如果检测出传输过程中的错误，主站重复发出询问。

2）管理、处理和发送以下可能的控制应用任务：将 4 个参数位发送给从站，例如设置门限值；改变从站的地址，如果从站支持这一特殊功能的话。

3）接入，新加入的 AS-i 从站被接入并存储到已检测到的从站表中，如果它们的地址不为 0，将被激活。主站如果处于保护模式，只有储存在主站的期望组态中的从站被激活。

第六节　点对点通信

点对点（Point to Point）通信简称为 PtP 通信。

一、点对点通信处理器与集成的点对点通信接口

1. CP 340 通信处理器

1 个通信接口，4 种不同型号：RS-232C（V. 24），20mA（TTY）和 RS-422/RS-485（X. 27），可以使用通信协议 ASCII、3964（R）和打印机驱动软件。

2. CP 341 通信处理器

通信协议包括 ASCII、3964（R）、RS 512 协议，和可装载的驱动程序，包括 MODBUS 主站协议或从站协议，和 Data Highway（DF1 协议）。

3. S7-300C 集成的点对点通信接口

全双工的传输速率为 19. 2kbit/s，半双工的传输速率为 38. 4kbit/s。

4. CP 440 点对点通信处理器

最多 32 个节点，传输速率最高为 115. 2kbit/s。可以使用的通信协议为 ASCII 和 3964（R）。

5. CP 441-11/CP 441-2 点对点通信处理器

CP 441-1 可以插入一块不同物理接口的 IF 963 子模块。

CP 441-2 可以插入两块分别带不同物理接口的 IF 963 子模块。

二、ASCII Driver 通信协议

PtP 通信协议如图 7-37 所示。

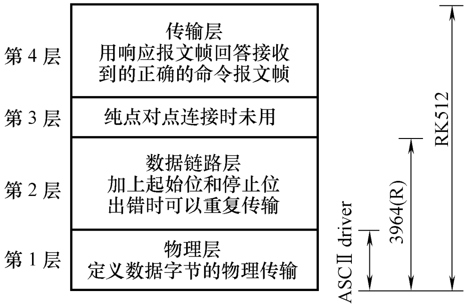


图 7-37 PtP 通信协议

1. 开放式的数据（所有可以打印的 ASCII 字符）**和所有其他的字符**

ASCII Driver 可以用结束字符、帧的长度和字符延迟时间作为报文帧结束的判据。用户可以在 3 个结束判据中选择一个。

（1）用结束字符作为报文帧结束的判据

用 1 ~ 2 个用户定义的结束字符表示报文帧的结束，应保证在用户数据中不包括结束字符。

（2）用固定的字节长度（1 ~ 1024B）作为报文帧结束的判据

如果在接收完设置的字符之前，字符延迟时间到，将停止接收，同时生成一个出错报文。接收到的字符长度大于设置的固定长度，多余的字符将被删除。接收到的字符长度小于设置的固定长度，报文帧将被删除。

（3）用字符延迟时间作为报文帧结束的判据

报文帧没有设置固定的长度和结束符，接收方在约定的字符延迟时间内未收到新的字符则认为报文帧结束（超时结束）。

2. 数据流控制/握手（Data Flow Control/Handshaking）

握手可以保证两个以不同速度运行的设备之间传输的数据。

（1）软件方式

例如通过向对方发送特定的字符（例如 XON/XOFF）实现数据流控制，报文帧中不允许出现 XON 和 XOFF 字符。

（2）硬件方式

例如用信号线 RTS/CTS 实现数据流控制，应使用 RS-232 C 完整的接线。

接收缓冲区已经准备好接收数据，就会发送 XON 字符或使输出信号 RTS 线为 ON。反之，如果报文帧接收完成，或接收缓冲区只剩 50B，将发送字符 XOFF，或使 RTS 线变为 OFF，表示不能接收数据。如果接收到 XOFF 字符，或通信伙伴的 CTS 控制信号被置为 OFF，将中断数据传输。如果在预定的时间内未收到 XON 字符，或通信伙伴的 CTS 控制信号为 OFF，将取消发送操作，并且在功能块的输出参数 STATUS 中生成一个出错信息。

3. CPU 31xC-2PtP 中的接收缓冲区

接收缓冲区是一个 FIFO（先入先出）缓冲区，如果有多个报文帧被写入接收缓冲区，总是第一个接收到的报文帧被传送到目标块中。如果想将最新接收的报文帧传送到目标块中，必须将缓存的报文帧个数设置为 1，并取消改写保护。

块校验字符（Block Check Characters，BCC）是正文中的所有字符“异或”运算的结果。这种校验方式又称为“纵向奇偶校验”。组态时可以选择报文的结束分界符中是否有 BCC。其具体计算如图 7-38 所示。

30H=0011 0000
31H=0011 0001
32H=0011 0010
10H=0001 0000
03H=0000 0011
XOR=0010 0000

图 7-38 块校验字符 BCC 计算

三、3964（R）通信协议

1. 3964（R）协议使用的控制字符与报文帧格式

3964（R）协议使用的控制字符见表 7-3。

BCC 是所有正文中的字符（包括正文中连发的 DLE）和报文帧结束标志（DLE 和 ETX）的“异或”运算的结果。

表 7-3　3964（R）协议使用的控制字符表

控制字符	数值	说　　明
STX	02H	被传送文本的起始点
DLE	10H	数据链路转换(Data Link Escape)或肯定应答
ETX	03H	被传送文本的结束点
BCC		块校验字符(Block Check Character),只用于 3964(R)
NAK	15H	否定应答(Negative Acknowledge)

正文中如果有字符 10H，在发送时自动重发一次。接收方在收到两个连续的 10H 时自动地剔除一个。

图 7-39 和图 7-40 分别为 3964（R）报文帧格式和 3964（R）报文帧传输过程。

SXT	正文(发送的数据)	DLE	ETX	BCC

图 7-39　3964（R）报文帧格式

发送方　接收方
开始代码(02H) —— STX → 建立连接
肯定应答(10H) ← DLE ——
数据字节 1 —— 字节 1 → 用户数据
数据字节 2 —— 字节 2 →
⋮　⋮
数据字节 n —— 字节 3 →
结束代码(10H) —— DLE → 释放连接
结束代码(03H) —— ETX →
块校验字符 —— BCC →
肯定应答(10H) ← DLE ——

图 7-40　3964（R）报文帧传输过程

2. 建立发送数据的连接

发送方首先应发送控制字符 STX。在“应答延迟时间（ADT）”到来之前，接收到接收方发来的控制字符 DLE，表示通信链路已成功地建立。

如果通信伙伴返回 NAK 或返回除 DLE 和 STX 之外的其他控制代码，或应答延迟时间到时没有应答，程序将再次发送 STX，重试连接。若约定的重试次数到后，都没有成功建立通信链路，程序将放弃建立连接，并发送 NAK 给通信伙伴，同时通过输出参数 STATUS 向功能块 P_SND_RK 报告出错。

接收方在接收到 DLE、ETX 和 BCC 后，根据接收到的数据计算 BCC，并与通信伙伴发送过来的 BCC 进行比较。如果两者相等，并且没有其他接收错误发生，接收方的 CPU 将发送 DLE，断开通信连接。如果两者不等，将发送 NAK，在规定的块等待时间内（4s）等待重新发送。如果在设置的重试次数内没有接收到报文，或者在块等待时间内没有进一步的尝试，将取消接收操作。

如果两台设备都请求发送。具有较低优先级的设备将暂时放弃其发送请求，向对方发送控制字符 DLE。具有较高优先级的设备将以上述方式发送其数据。等到高优先级的传输结束，连接被释放，具有较低优先级的设备就可以执行其发送请求。通信的双方必须设置优先级。

四、用于 CPU 31xC-2PtP 点对点通信的系统功能块

CPU 31xC-2PtP 点对点通信的系统功能块见表 7-4。

表 7-4　CPU 31xC-2PtP 点对点通信系统功能块

系统功能块		说　　明
SFB 60	SEND_PTP	将整个数据块或部分数据块区发送给一个通信伙伴
SFB 61	RCV_PTP	从一个通信伙伴接收数据,并将它们保存在一个数据块中
SFB 62	RES_RCVB	复位 CPU 的接收缓冲区
SFB 63	SEND_RK	将整个数据块或部分数据块区发送给一个通信伙伴
SFB 64	FETCH_RK	从一个通信伙伴处读取数据,并将它们保存在一个数据块中
SFB 65	SERVE_RK	从一个通信伙伴处接收数据,并将它们保存在一个数据块中; 为通信伙伴提供数据

1. 用 SFB60“SEND_PTP”发送数据（ASCII/3964（R））

块被调用后，在控制输入 REQ 的脉冲上升沿发送数据。SD_1 为发送数据区（数据块编号和起始地址），LEN 是要发送的数据块的长度。用参数 LADDR 声明在 HW Config（硬件组态）中指定的子模块的 I/O 地址。在控制输入 R 的脉冲上升沿，当前的数据发送被取消，SFB 被复位为基本状态。被取消的请求用一个出错报文（STATUS 输出）结束。如果块执行没有错误，DONE 被置为 1 状态。如果出错，ERROR 被置为 1 状态，STATUS 将显示相应的事件标符（ID）。如果块被正确执行后 DONE 为 1，则意味着：

1）使用 ASCII Driver 时，数据被传送给通信伙伴，但是不能保证被对方正确地接收。

2）使用 3964（R）协议时，数据被传送给通信伙伴，并得到对方的肯定确认。但是不能保证数据被传送给对方的 CPU。SFB 最多只能发送 206 个连续的字节。必须在参数 DONE 被置为 1，发送过程结束时，才能向 SD_ 1 指定的发送区写入新的数据。

2. 用 SFB 61“RCV_PTP”接收数据

SFB 61 用来接收数据，并将它们保存到一个数据块中。调用 SFB 61 后，令控制输入 EN_ R 为 1，接收数据的准备就绪。EN_ R 的状态为 0，接收操作就被闭锁。RD_ 1 为接收区，LEN 是数据块的长度。块被正确执行时 NDR 被置为 1 状态。如果请求因出错被关闭，ERROR 被置为 1 状态。如果出现错误或报警，STATUS 将显示相应的事件标识符（ID）。

3. 用 SFB 62“RES_RCVB”清空接收缓冲区

SFB 62 用于清空 CPU 的整个接收缓冲区。在调用 SFB 62 时接收到的报文帧将被保存。

通信协议系统功能块如图 7-41 所示。

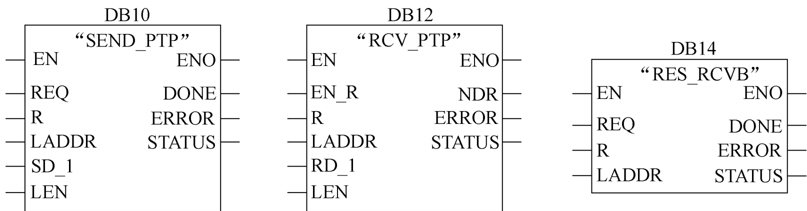


图 7-41　通信协议系统功能块

第七节　工业以太网

工业以太网是为工业应用专门设计的，它是遵循国际标准 IEEE 802. 3（Ethernet）的开放式、多供应商、高性能的区域和单元网络。工业以太网已经广泛地应用于控制网络的最高层，并且有向控制网络的中间层和底层（现场层）发展的趋势。

一、工业以太网介绍

1. 以太网的特点

企业内部互联网（Intranet）、外部互联网（Extranet）以及国际互联网（Internet）不但进入了办公室领域，而且已经广泛地应用于生产和过程自动化。继 10Mbit/s 以太网成功运行之后，具有交换功能、全双工和自适应的 100Mbit/s 高速以太网（Fast Ethernet）也已经成功运行多年。SIMATIC NET 可以将控制网络无缝集成到管理网络和互联网。以太网的市场占有率高达 80%，毫无疑问是当今局域网（LAN）领域中首屈一指的网络。以太网有以下优点：

1）可以采用冗余的网络拓扑结构，可靠性高。

2）通过交换技术可以提供实际上没有限制的通信性能。

3）灵活性好，现有的设备可以不受影响地扩张。

4）在不断发展的过程中具有良好的向下兼容性，保证了投资的安全。

5）易于实现管理控制网络的一体化。

以太网可以接入广域网（WAN），例如互联网，可以在整个公司范围内通信，或实现公司之间的通信。SIMATIC NET 供应的节点已超过 400000 个，可以用于严酷的工业环境，包括有强烈电磁干扰的区域。

2. 工业以太网的构成

典型的工业以太网由以下 4 类网络器件组成：

1）连接部件：包括 FC 快速连接插座、电气链接模块（ELS）、电气交换模块（ESM）、光纤交换模块（OSM）和光纤电气转换模块（MC TP11）。

2）通信介质：可以采用普通双绞线、工业屏蔽双纹线和光纤。

3）SIMATIC PLC 的工业以太网通信处理器：用于将 PLC 连接到工业以太网。

4）PG/PC 的工业以太网通信处理器：用于将 PG/PC 连接到工业以太网。

二、工业以太网的网络方案

工业以太网可以采用下面的 3 种方案。

1. 同轴电缆网络

网络以三同轴电缆作为传输介质，由若干条总线段组成，每段的最大长度为 500m。一条总线段最多可以连接 100 个收发器，可以通过中继器接入更多的网段。

网络为总线型结构，因为采用了无源设计和一致性接地的设计，极其坚固耐用。网络中各设备共享 10Mbit/s 带宽。

可以混合使用电气网络和光纤网络，使两者的优势互补，网络的分段改善了网络的性能。三同轴电缆网络有分别带一个或两个终端设备接口的收发器，中继器用来将最长 500m 的分支网段接入网络中。

2. 双绞线和光纤网络

双绞线和光纤网络的传输速率为 10Mbit/s，可以是总线型或星形拓扑结构，使用光纤连接模块（OLM）和电气链接模块（ELM）。

OLM 和 ELM 是安装在 DIN 导轨上的中继器，它们遵循 IEEE 802.3 标准，带有 3 个工业双绞线接口，OLM 和 ELM 分别有两个和一个 AUI 接口。在一个网络中最多可以级联 11 个 OLM 或 13 个 ELM。

3. 高速工业以太网

高速工业以太网的传输速率为 100Mbit/s，使用光纤交换模块（OSM）或电气交换模块（ESM）。工业以太网与高速工业以太网的数据格式、CSMA/CD 访问方式和使用的电缆都是相同的，高速以太网最好用交换模块来构建。

4. 以太网和高速以太网的工作过程

以太网使用带冲突检测的载波侦听多路访问（CSMA/CD）协议，各站用竞争方式发送信息到传输线上，两个或多个站可能因同时发送信息而发生冲突。为了保证正确地处理冲突，以太网的规模必须根据一个数据包最大可能的传输延迟来加以限制。在传统的 10Mbit/s 以太网中，允许的冲突范围为 4520m，因为传输速率的提高，高速以太网的冲突范围减小为 452m。为了扩展冲突范围，需要使用有中继器功能的网络部件，例如工业以太网的 OLM 和 ELM。用具有全双工功能的交换模块来构建较大的网络时，不必考虑高速以太网冲突区域的减小。

三、工业以太网的交换技术

1. 交换技术

在共享局域网（LAN）中，所有站点共享网络性能和数据传输带宽，所有的数据包都经过所有的网段，在同一时间只能传送一个报文。

在交换式局域网中，每个网段都能达到网络的整体性能和数据传输速率，在多个网段中可以同时传输多个报文。本地数据通信在本网段进行，只有指定的数据包可以超出本地网段的范围。交换模块是从网桥发展而来的设备，利用终端的以太网 MAC 地址，交换模块可以对数据进行过滤，局部子网的数据仍然是局部的，交换模块只传送发送到其他子网络终端的数据。与一般的以太网相比扩大了可以连接的终端数，可以限制子网内的错误在整个网络上的传输。虽然较复杂，但交换技术与中继技术相比有以下优点：

1）可以选择用来构建部分网络或是网段，通过数据交换结构提高了数据吞吐量和网络性能，网络配置规则简单。

2）不必考虑传输延时，可以方便地实现有 50 个 OSM 或 ESM 的网络拓扑结构。通过连接单个的区域或部分网络，可以实现网络规模的无限扩展。

2. 全双工模式

在全双工模式，一个站能同时发送和接收数据。如果网络采用全双工模式，不会发生冲突。全双工模式需要采用发送通道和接收通道分离的传输介质，以及能够存储数据包的部件。由于在全双工连接中不会发生冲突，支持全双工的部件可以同时以额定传输速率发送和接收数据，因此以太网和高速以太网的传输速率分别提高到 20Mbit/s 和 200Mbit/s。由于不需要检测冲突，全双工网络的距离仅受到它使用的发送部件和接收部件性能的限制，使用光纤网络时更是如此。

第八章　PLC工程应用开发

第一节　工程设计原则

在设计 PLC 控制系统时，应遵循以下基本原则。

1. 满足要求

任何一种控制系统都是为了实现被控对象的工艺要求，以提高产品质量和生产效率。因此，充分发挥 PLC 的功能，最大限度地满足被控对象的控制要求，是设计控制系统的首要前提，这也是工程设计中最重要的一条原则。这就要求设计人员在设计前就要深入现场进行调查研究。收集控制现场的资料，收集控制过程中有效的控制经验，进行系统设计。同时注意要和现场的管理人员、技术人员、工程操作人员紧密配合，结合生产工艺，拟定控制方案，共同解决设计中的重点问题和疑难问题。

2. 安全可靠

控制系统长期运行中能否达到安全、可靠、稳定，是设计控制系统的重要原则。为了能达到这几点，要求在系统设计、器件选择和软件编程上都要全面考虑。比如说，在硬件和软件的设计上，应该保证 PLC 程序不仅在正常条件下能正确运行，而且在一些非正常情况下（如突然掉电再上电，按钮按错等）也能正常工作。程序只能接受合法操作，对非法操作程序能予以拒绝等。

3. 经济实用

一个新的控制工程固然能提高产品的质量，提高产品的生产效率，从而为工程带来巨大的经济效益和社会效益。但是，新工程的投入、技术的培训、设备的维护也会导致工程的投入和运行资金的增加。在满足控制要求的前提下，一方面要注意不断地扩大工程的效益，另一方面也要注意不断地降低工程的运行成本。这就要求，不仅应该使控制系统简单、经济，而且要使控制系统的使用和维护既方便又低成本。

4. 适应发展

社会在不断地前进，科学在不断地发展，控制系统的要求也一定会不断地提高、不断地完善。因此，在设计控制系统时要考虑到今后的发展、完善，避免重复投资。这就要求在选择 PLC 机型和 I/O 模块时要能适应发展的需要，要适当留有余量。PLC 系统工程设计流程图如图 8-1 所示。

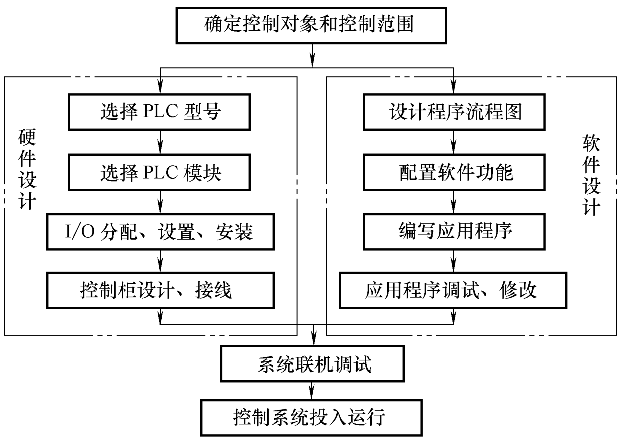


图 8-1　PLC 系统工程设计流程图

第二节　需求分析

需求分析是指用户对需要开发的控制系统在设计、功能、性能、工艺等方面的要求，这是直接关系到应用系统性能、成本、经济效益等指标的关键环节。在一个 PLC 应用系统中，明确其控制对象和控制范围是很重要的。需求分析阶段要对被控制生产过程进行深入的分析，对被控设备进行分析，了解被控设备的组成结构、驱动方式等，明晰用户的需求。工程开发设计人员应该深入现场，全面了解被控制的生产过程、工艺过程和环境特点以及对控制的要求等。例如，各个生产过程或设备的起停条件，它们之间是否有联锁条件，时间和数量上有什么关系，对重要的参数是否需要冗余或采用何种冗余方式等。需求分析的基本任务是：用户和 PLC 工程开发人员双方一起来充分地理解用户的要求，并把双方共同的理解明确表达出来。

第三节　硬件设计

一、PLC 机型选择

1. 合理的结构类型

PLC 主要有整体式和模块式两种结构类型。整体式 PLC 的每一个 I/O 点的平均价格比模块式的便宜，且体积相对较小，一般用于控制系统工艺过程较为固定、环境条件较好、维修较少的小型控制系统中。而模块式 PLC 的功能扩展灵活方便，在 I/O 点数、输入点数与输出点数的比例、I/O 模块的种类等方面选择余地大，而且维护方便，一般用于较复杂的控制系统。

2. 安装方式的选择

PLC 系统的安装方式分为集中式、远程 I/O 式以及多台 PLC 联网的分布式。集中式不需要设置驱动远程 I/O 硬件，系统反应快、成本低。远程 I/O 式适用于大型系统，系统的装置分布范围很广，远程 I/O 可以分散安装在现场装置附近，连线短，但需要增设驱动器和远程 I/O 电源。多台 PLC 联网的分布式适用于多台设备分别独立控制，又要相互联系的场合，可以选用小型 PLC，但必须要附加通信模块，将多台 PLC 联成网络系统。

3. 相应的功能要求

根据控制系统功能的强弱适当选型，选择最合适的。一般小型（低档）PLC 具有逻辑运算、定时、计数等功能，可以满足只需要开关量控制、控制速度要求不高的设备，如 S7-200 系列。

对于以开关量控制为主，带少量模拟量控制的系统，可选用能带 A-D 和 D-A 转换单元，具有加减算术运算、数据传送功能的增强型中、低档 PLC，如 S7-300 系列。

4. 响应速度要求

响应时间是指从外部信号输入到给定输出的总时间，通常包括：输入滤波器的延迟时间、I/O延迟时间、逻辑运算时间、输出元件的延迟时间。扫描方式引起的延迟可达到 2～3 个扫描周期。所以，PLC 存在 I/O 响应延迟，因此在快速响应设备中应加以注意。

5. 系统可靠性的要求

对于一般系统 PLC 的可靠性均能满足。对可靠性要求很高的系统，应考虑是否采用冗余系统或热备用系统。

6. 机型尽量统一

一个大型企业在选用 PLC 时，应尽可能做到 PLC 的机型统一。主要因为：

1）控制系统中的机型统一，其外部设备通用，资源可共享，易于联网通信，连接上位机后

易于形成一个多级分布式控制系统。

2）使控制系统中的模块功能和使用方法类似，有利于使用人员熟练操作，有利于使用人员的技术培训和技术水平的提高。

3）使控制系统中的模块可互为备用，便于备品备件的采购和管理。

二、确定容量参数

1. 对 I/O 点数的估计

对开关量输入，按参数等级分类统计。对开关量输出，按输出功率要求及其他参数分类统计。对模拟量 I/O，按点数进行粗估。

I/O 点数的确定要充分考虑冗余，由于采用的控制方法或编程水平不一样，I/O 点数可能不同。根据工程经验，通常 I/O 点数按实际工程需要的 15% ~30% 考虑余量。

（1）开关量点数

开关量控制是 PLC 的基本功能，对于开关量控制系统，主要考虑 PLC 的最大开关量点数是否能满足系统的要求。常用 I/O 控制设备的 I/O 控制点见表 8-1。

（2）模拟量点数

对于要求模拟量控制的系统，主要考虑 PLC 的最大模拟量点数是否能满足系统的要求，及每个模块的点数。

表 8-1　常用 I/O 控制设备的 I/O 控制点

I/O 控制设备	输入点数	输出点数	I/O 控制设备	输入点数	输出点数
行程开关	1		风机		1
接近开关	1		单控电磁阀	2	1
位置开关	2		双控电磁阀	3	2
光电开关	2		单向运行直流电动机	9	6
按键	1		可逆运行直流电动机	12	8
信号灯		1			

2. 对 PLC 性能要求的估计

需要考虑：是否有特殊控制功能要求（如高速计数器等）？机房离现场的最远距离为多少？现场对控制器响应速度有何要求？

在此基础上，选择 PLC 时还需注意以下两个问题。

（1）PLC 可带 I/O 点数

有的产品手册或产品目录单上给出的最大输入点数或最大输出点数，常意味着只插输入模块或只插输出模块的容量，即实际给出的是输入、输出容量之和，有时也称为扫描容量。在实际工程设计时，通常不能按其最大输入点数或最大输出点数来设计。

（2）PLC 通信距离和速度

产品手册上给出的覆盖距离，也称为最大距离，包括远程 I/O 模块在内可以达到的距离。但远程 I/O 模块的 I/O 反应速度大大下降，一般为 19.2k 波特率。

3. 对所需内存容量的估计

内存容量是 PLC 本身能提供的硬件存储单元大小，程序容量是存储器中用户应用项目使用的存储单元的大小，因此程序容量小于存储器容量。

用户程序所需内存与下列因素有关：

1）逻辑量 I/O 点数的估计。

2）模拟量 I/O 点数的估计。

3）内存利用率的估计。

4）程序编制者的编程水平的估计。

三、系统软、硬件选择

1. 硬件选择

（1）CPU 的选择

CPU 的选择是合理配置系统资源的关键，选择时必须考虑控制系统对 CPU 的要求，包括系统集成功能、程序块数量限制、各种位资源、MPI 接口能力、是否有 PROFIBUS-DP 主从接口、RAM 容量、温度范围等。最好在西门子公司的技术支持下进行，以获得合理的、最佳的选择方案。

（2）扩展方式选择

S7-300 PLC 有多种扩展方式，实际选用时，可通过控制系统接口模块扩展机架、PROFIBUS-DP 现场总线、通信模块、远程 I/O 及 PLC 子站等来扩展 PLC 或预留扩展口。

（3）PLC 的联网

PLC 的联网包括 PLC 与计算机联网和 PLC 之间相互联网两种方式。因为 S7-300 PLC 的工业通信网络淡化了 PLC 与 DCS 的界限，联网的解决方案很多，用户可根据企业的要求选用。通常需要考虑 PLC 有什么类型的通信接口，最多可以配置多少个接口，可以使用什么通信规约，通信的速率是多少，最远通信距离是多少等。

PLC 系统的通信网络中，网络通信负荷应不大于60%。PLC 系统的通信网络的主要形式有下列几种：

1）PC 为主站，多台同型号 PLC 为从站，组成简易 PLC 网络。

2）1 台 PLC 为主站，其他同型号 PLC 为从站，构成主从式 PLC 网络。

3）PLC 网络通过特定网络接口连接到大型 DCS 中作为 DCS 的子网。

4）专用 PLC 网络（各厂商的专用 PLC 通信网络）。

（4）系统的可靠性

对可靠性要求极高的系统，在装置选型时，还应考虑 PLC 系统是否有冗余功能，是否能构成冗余系统。包括下列两个方面：

1）控制单元的冗余：重要的过程单元，如 CPU（包括存储器）及电源均可以 1:1 冗余。

2）I/O 接口单元的冗余：重要检测点的 I/O 模块可冗余配置。

2. 软件的选择

（1）编程软件的选择

这主要考虑对 CPU 的支持状况，STEP 7 的早期版本对有些型号的 CPU 不支持，硬件组态时会发生故障，而 STEP 7 V5.4 则不存在这种问题。

（2）PLC 指令系统

对于简单控制系统，一般的小型 PLC 即可以满足功能。如果系统要求 PLC 实现某些特殊功能，在装置选型时应考虑是否有相应的指令系统支持，还应考虑 PLC 的编程元件（定时器、计数器、数据寄存器、辅助继电器等）是否够用。当然，通常情况这些是够用的。

第四节　软 件 设 计

一、控制程序的设计

1. 控制程序设计的基本要求

控制程序设计的基本要求是由 PLC 自身的特点及其在工业控制过程中要求实现的具体控制

功能来决定的。

(1) 紧密结合生产工艺过程

每个控制系统都是为了实现对一定的生产过程的控制而设计的。不同的生产工艺过程要求有不同的控制功能。控制程序的设计人员必须对被控制生产过程进行深入的分析，具体了解用户的实际需求，了解被控制的生产过程、工艺过程和环境特点以及对控制的要求等。所以，控制程序的设计人员必须严格按照生产工艺的具体要求来设计控制程序。

(2) 熟悉控制产品的硬件结构

不同的硬件结构决定着不同的控制软件系统。所以，控制程序的设计人员不能抛开硬件结构而单独地考虑软件系统，要结合硬件结构来编制控制程序。

(3) 兼有计算机和自动控制两方面的知识

PLC 是以微处理器为核心的控制设备。可以说，PLC 技术在一定程度上是从计算机技术和自动控制技术发展演变而来的。所以，控制程序设计人员必须兼有计算机和自动控制两方面的技术知识。

2. 控制程序设计的基本原则

控制程序设计是以系统要实现的工艺要求、硬件组成和操作方式等条件为依据进行的，程序设计一般应遵循的原则是：逻辑关系简单清晰，易于编程输入，少占内存，减少扫描时间。具体来说，有以下几点原则。

1) 控制程序设计一般只需要考虑用户程序的设计，而由系统自动实现外围设备的管理，控制程序的设计人员只是设置必要的参数。

2) 对 I/O 信号进行统一配置和操作，确定各个信号在一个扫描周期内的唯一状态，避免由于状态的不同而引起的逻辑混乱。

3) 由于 CPU 在每个周期内都固定进行某些窗口服务，占用一定的机器时间，所以，周期时间不能无限制地缩短。

4) 定时器的时间设定值不能小于周期扫描时间，而且，当定时器时间设定值不是平均周期时间的整倍数时，可能会有定时误差。

5) 用户程序中如果多次对同一个参数进行赋值，则只有最后一次操作有效，前几次操作不影响实际输出值。

3. 控制程序的设计方法

控制程序设计，有以下几种方法。

(1) 时序流程图法

时序流程图法是首先画出控制系统的时序图，再根据时序关系画出对应的控制任务的程序框图，最后把程序框图写成 PLC 程序。时序流程图法很适合于以时间为基准的控制系统的编程。

(2) 步进顺控法

一般比较复杂的程序，都可以分成若干个功能比较简单的程序段，一个程序段可以看成是整个控制过程中的一步。从这个角度去看，一个复杂的系统的控制过程是由这样若干个步组成的。系统控制的任务实际上可以认为是在不同时刻或者在不同进程中去完成对各个步的控制。

(3) 经验法

经验法是运用自己的或别人的经验，例如在一些典型电路的基础上进行设计。多数是设计前，先选择与自己工艺要求相近的程序，结合自己工程的情况，根据被控对象对控制系统的具体要求，对这些“试验程序”不断修改、调试，使之适合自己的工程要求。这里所说的经验，有

的是来自自己的经验总结，有的可能是别人的设计经验。

（4）计算机辅助设计

计算机辅助设计是通过 PLC 编程软件在计算机上进行程序设计、离线或在线编程、离线仿真和在线调试等。编程软件 STEP 7 和 WinCC，仿真软件 PLCSIM 等都是 S7 系列 PLC 编程常用软件。使用这些编程软件可以十分方便地在计算机上离线或在线编程、在线调试。

4. 控制程序的设计过程

（1）对系统任务分块

在 PLC 应用系统中，程序是由一些基本环节组成的。分块的目的就是把一个复杂的工程分解成多个比较简单的小任务。这样就把一个复杂的、大的问题化为多个简单的、小的问题，便于编制程序。

（2）编写控制系统的逻辑关系图

从逻辑关系图上可以反映出某一逻辑关系的结果是什么，这一结果又应该导出哪些动作。这个逻辑关系可以以各个控制活动顺序为基准，也可能以整个活动的时间节拍为准。逻辑关系图反映了输入与输出的关系。

（3）绘制各种电路

在绘制 PLC 的输入电路时，要考虑到输入端的电压和电流是否合适，也要考虑到在特殊条件下运行的可靠性与稳定条件等问题。特别要考虑到是否会把高压引导到 PLC 的输入端，这样会对 PLC 造成比较大的损坏。

（4）编写 PLC 程序并进行模拟调试

控制程序是整个控制系统工作的核心，是保证系统工作正常、可靠的关键。所以，控制程序必须经过反复调试、修改，直到符合控制工艺要求。

（5）制作控制柜

对于安装在室内的控制柜一般符合 IP52 标准，对于安装在室外的控制柜一般符合 IP56 标准。在控制柜的内部提供 AC 220V 标准插座，在柜门内侧有电源开关。端子排、电缆走线槽均由阻燃材料制造。端子排的安装位置应便于接线，距柜底不小于 300mm，距柜顶不小于 150mm，排与排之间的距离不小于 200mm。

（6）现场调试

控制程序的控制对象是现场的设备，所以，编写好的控制程序必须要通过设备的现场联调。

（7）现场试运行并编写技术文件

经过现场调试以后，控制电路和控制程序基本被确定了。这时就要全面整理技术文件，包括整理电路图、PLC 程序、使用说明及帮助文件。

二、控制系统的设计

1. 开关量控制系统的设计

开关量控制是指控制系统的输入信号和输出信号都是只有两个状态的开关量。这类系统包含手动、单次和自动控制。这类系统的设计要特别注意 I/O 模块的隔离、接口的匹配和功率的消耗问题。

（1）手动控制

手动控制在调试、维修过程中是不可少的。

（2）单次控制

这种控制的特点是一旦控制系统被起动之后，控制过程将自动完成一个周期。如果系统需要再次起动，则必须再次人工起动。这种系统更便于参数的修改、调整。

（3） 自动控制

系统起动之后，就可以按照工程要求进行控制。整个控制过程无人工干预。系统对输入/输出要求都很严格，系统的可靠性、安全性设计尤为重要。

2. 模拟量控制系统的设计

模拟量控制系统是指输入信号为模拟量的控制系统。控制系统的控制方式可分为开环控制和闭环控制。闭环控制根据其设定值的不同，又可分为调节系统和随动系统两种。

1）调节系统的设定值是由控制系统的控制器给出，控制器的作用就是使反馈值向给定值靠近，以反馈值对设定值的偏差最小为目的。

2）随动系统的设定值是由被控制对象给出的，控制器的作用就是使控制目标不断地向被控对象靠近。各种跟踪系统都是随动系统。

在模拟量控制系统中，信号的屏蔽非常关键，一般可采取屏蔽电缆传送模拟信号。注意对多个模拟信号共用一根多芯屏蔽电缆或用两种屏蔽电缆传送时，信号间一定要做好屏蔽。而且电缆的屏蔽层一端（一般在控制柜端）要可靠接地。

模拟量控制系统的设计中特别要注意抗干扰问题。解决干扰的办法有 4 个：

1）接地问题。接地就是接大地，包括 PLC 接地端的接地，要真接地，不要假接地。

2）模拟信号线的屏蔽向题。屏蔽线的始端和终端都要接地。信号线的屏蔽是防止干扰的重要措施。

3）对某些高频信号要解决匹配问题。如果不匹配，很容易在信号传送中引进干扰，使信息失真。

4）对信号进行滤波。

第五节　系统调试

1. 系统调试内容

系统调试包括硬件调试、下载用户程序、系统功能的测试、记录对程序的修改、保存和压缩程序等。

可以通过“监视/修改变量”工具来调试硬件。

下载用户程序之前应执行 CPU 存储器的复位并将 CPU 切换到 STOP 状态。用户程序中应包含硬件的组态数据。

检查系统的功能是否正常。如果用户程序是结构化的程序，可以在组织块中逐一调用各程序块，一步一步地调试程序。

必须记录调试过程中对程序的修改。可采取的最简单的方法是在程序清单上手工记录修正内容，也可以给块加上适当的注释或调整版本号以反映修改。

调试结束后，应保存最终版本的程序。

2. 装载用户程序

在下装新的全部用户程序之前，应该执行一次 CPU 存储器的复位。为安全起见，应该在停机状态下执行下载。下装时，一次可以把个别块、几个块或全部的程序下载到 CPU。具体操作如下：

1）当选择 S7 程序的文件夹时就选择全部用户程序。

2）用鼠标选择个别的块。

3）按住“Ctrl”键并用鼠标选择几个块。另一个方法是按住“Shift”键，并选择第一个块

和最后一个块，或者用鼠标框选中要选择的块。

单击图标，起动块的下载，下载用户程序到 CPU 如图 8-2 所示。

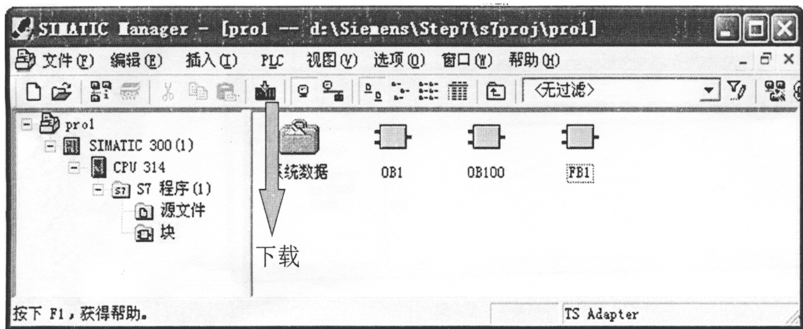


图 8-2　下载用户程序到 CPU

3. 排除停机错误

如果没有编写错误处理组织块或错误处理组织块中调用 SFC“STOP”，当程序出现错误或硬件出现故障时 CPU 就进入停机状态。利用诊断缓冲区可以确定停机的原因。

诊断缓冲区是存放在 CPU 中的一个先进先出缓冲区，它由后备电池来保持，对存储器的复位也不能清除该缓冲区。它存储按照发生顺序排列的诊断事件。所有的事件可以在编程器上按照它们出现的顺序进行显示。

当选择一个事件时，在“事件详细内容”窗口中出现附加的一些信息：除了事件标识和事件号，还有有关事件的附加信息，例如：出现事件的指令地址等。

4. 系统功能的测试

系统功能的测试是指一步一步地调试结构化程序，每个块包含特定的系统功能。第一步是通过下载起动组织块（OB 100 ~ OB 102）来测试起动特性。然后，一步一步地测试循环程序。从嵌套最深的块（例如：FB4）开始调试。这样，需要在 OB1 中插入一个块调用指令。然后，可以调试系统功能，它包括若干个块（例如 FC 1）。为此，在 OB1 中插入一个带有 BEU 指令的段。当所有的程序都被调用后，再删除这个段。根据程序的结构，用于中断处理的程序或在最后测试（如果该中断程序不影响程序的循环执行），或在循环程序的测试过程中调试。系统功能测试的步骤如图 8-3 所示。

5. 记录程序的修改

在调试中有不同的方法来记录程序的修改。

块编辑器提供不同的注释功能。新插入的段，应该在块注释中说明。在相关的段注释中应该包括段的修改记录和段的功能说明。当用 STL 语言编写程序时，可以对每条指令说明或在指令之间说明。

项目、S7 程序、块的“对象属性”提供额外的说明功能。用鼠标右键选择对象后，选择“属性”菜单选项，在“属性”中，输入有关修改的附加说明。在块属性中，有版本标识、块名称、系列和作者的输入区域。

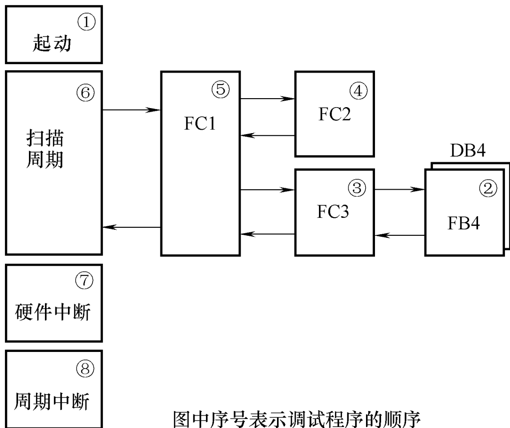


图 8-3　系统功能测试的步骤

第六节　可靠性设计

一、影响现场输入给 PLC 信号出错的主要原因

1）由于机械拉扯或线路自身老化等原因，造成传输信号线短路或断路，当传输信号线出现故障时，现场信号无法传送给 PLC，造成控制出错。

2）机械触点抖动，现场触点虽然只闭合一次，PLC 却认为闭合了多次。虽然硬件增加了滤波电路，软件增加了微分指令，但由于 PLC 扫描周期太短，仍可能在计数、累加、移位等指令中出错，出现错误控制结果。

3）现场变送器、机械开关自身出故障，如触点接触不良，变送器反映现场非电量偏差较大或不能正常工作等，这些故障同样会使控制系统不能正常工作。

二、影响执行机构出错的主要原因

1）控制负载的接触不能可靠动作，PLC 发出了动作指令，但执行机构并没按要求动作。

2）控制变频器起动，由于变频器自身故障，变频器所带电动机并没按要求工作。

3）各种电动阀、电磁阀该开的没能打开，该关的没能关到位。

例如，来自电源线的杂波，能造成系统电压畸变，导致系统内电气设备过电压、过负荷、过热甚至烧毁元器件，造成 PLC 等控制设备误动作。通常采取的措施是，在电源入口处最好设置屏蔽变压器或电源滤波等防干扰设施。其中，电源滤波器的接地要以最短的线路接到中央保护地。对于直流电源，则可加装微分电容加以干扰抑制。

三、硬件可靠性设计

1. PLC 的安装

（1）PLC 安装的一般性指导

PLC 安装需要注意以下几方面：

1）在对 PLC 接线时要确保所有的电器符合国家和地区的电气标准。及时同地区的权威机构保持联系，以确定哪些标准符合你的特殊需要。

2）要正确地使用导线。

3）不要将连接器的螺钉拧得过紧。

4）尽可能使用短导线（最长 500m 屏蔽线，或 300m 非屏蔽线），导线要尽量成对使用，用一根中性或公共导线与一根热线或信号线相配对。

5）将交流线和大电流快速开关的直流线与小电流的信号线隔开。

6）正确地识别和划分 PLC 模块的接线端子。

7）针对闪电式浪涌，安装合适的浪涌抑制设备。

8）控制设备在不安全条件下可能会失灵，导致被控制设备的误操作。这样的误动作会导致严重的人身伤害和设备严重损坏。可以考虑使用独立于 PLC 的紧急停机功能、机电过载保护设备，或其他冗余保护。

（2）使用隔离电路时的接地与电路参考点指南

使用隔离电路时的接地与电路参考点应遵循以下几点：

1）应该为每一个安装电路选一个参考点（0V），这些不同的参考点可能会连在一起，这种连接可能会导致预想不到的电流，它们会导致逻辑错误或损坏电路。产生不同参考电动势的原因，经常是由于接地点在物理区域上被分隔的太远。当相距很远的设备被通信电缆或传感器连接起来的时候，由电缆线和地之间产生的电流就会流经整个电路。即使在很短的距离内，大型设备

的负载电流也可以在其与地电动势之间产生变化，或者通过电磁作用直接产生不可预知的电流。那些没有正确选定参考点的电源，相互之间的电路中有可能产生毁灭性的电流，以致破坏设备。

2）当把几个具有不同电位的 CPU 连到一个 PPI 网络时，应该采用隔离的 RS-485 中继器。

PLC 产品已在特定点上安装了隔离元件，以防止安装中有不期望的电流产生。如果打算安装，应考虑哪些地方有这些隔离元件，哪些地方没有。同时也应考虑到相关电源之间的隔离以及其他设备的隔离，还有相关电源的参考点都在什么地方。

3）最好选择一个接地参考点，并且用隔离元件来破坏可能产生不可预知电流的、无用的电流回路。在暂时性连接中可能引入新的电路参考点，比如编程设备与 CPU 连接的时候。

4）在现场接地时，一定要随时注意接地的安全性，并且要正确地操作隔离保护设备。

5）在大部分的安装中，如果把传感器的供电 M 端子接到地上可以获得最佳的噪声抑制效果。

（3）S7-300 隔离特性

S7-300 隔离特性如下：

1）CPU 逻辑参考点与 DC 传感器提供的 M 点类似。

2）CPU 逻辑参考点与采用 DC 电源供电的 CPU 输入电源提供的 M 点类似。

3）CPU 通信端口与 CPU 逻辑口（DP 口除外），具有同样的参考点。

4）模拟输入及输出与 CPU 逻辑不隔离，模拟输入采用差动输入并提供低压公共模式的滤波电路。

5）逻辑电路与地之间的隔离为 AC 500V。

6）DC 数字输入和输出与 CPU 逻辑之间的隔离为 AC 500V。

7）DC 数字 I/O 组的点之间隔离为 AC 500V。

8）继电器输出、AC 输出和输入与 CPU 逻辑之间的隔离为 AC 1500V。

9）继电器输出组的点之间隔离为 AC 1500V。

10）AC 电源线和零线与地、CPU 逻辑以及所有的 I/O 之间的隔离为 AC 1500V。

2. 电源的安装

（1）交流输入 PLC 的安装

AC 交流接线安装时的一般性指南如下。

1）用一个单刀开关将电源与 CPU、所有的输入电路和输出（负载）电路隔离。

2）用一台过电流保护设备保护 CPU 的电源、输出点以及输入点。也可以为每个输出点加上熔丝进行范围更广的保护。

3）将 S7-300 的所有地线端子和最近接地点相连接，以获得最好的抗干扰能力。建议使用 1.50mm^2 的电线连接到独立导电点上（也称为一点接地）。

4）本机单元的直流传感器电源可用作本机单元的输入和扩展 DC 输入以及扩展继电器线圈供电，这一传感器电源具有短路保护功能。

5）在大部分的安装中，如果把传感器的供电 M 端子接到地上可以获得最佳的噪声抑制效果。

（2）直流输入 PLC 的安装

DC 隔离安装接线的一般性指南如下。

1）用一个单刀开关将电源同 CPU、所有的输入电路和输出（负载）电路隔离开。

2）用过电流保护设备保护 CPU 电源、输出点以及输入点。也可以在每个输出点加上熔丝进行过电流防护。

3）确保 DC 电源有足够的抗冲击能力，以保证在负载突变时，可以维持一个稳定的电压，这时需要一个外部电容。

4）在大部分的应用中，把所有的 DC 电源接到地可以得到最佳的噪声抑制效果。在未接地 DC 电源的公共端与保护地之间接以电阻与电容并联电路。电阻提供了静电释放通路，电容提供了高频噪声通路，它的典型值是 4700pF。

5）DC 24V 电源回路与设备之间，以及 AC120/230V 电源与危险环境之间，必须提供安全电气隔离。

3. 抑制电路的使用

（1）抑制电路使用的一般性指导

在感性负载中要加入抑制电路，以抑制在关闭电源时电压的升高。可以采用下面的方法来设计具体的抑制电路。设计的有效性取决于实际的应用，因此必须调整参数以适应具体的应用。要保护所有的器件参数与实际应用相符合。

（2）继电器输出模块的保护

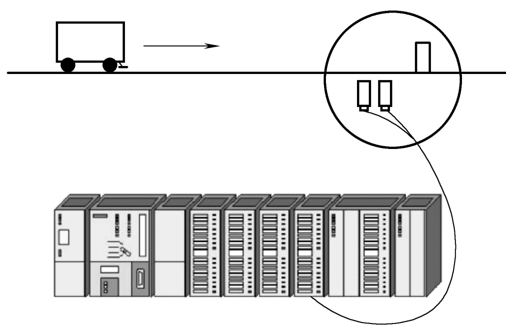
对继电器输出模块的保护主要有两个方面：一个方面是对继电器触点的保护，使电感在断电时不会产生高压加到继电器的触点；另一个方面是对电源的保护，使为继电器提供电压的电源不受高电压的冲击。抑制高电压的主要办法是在感性负载两端并联 RC 吸收电路，对交流电源除了用 RC 电路吸收之外，还可以并联电阻以消除电压冲击。

4. 保护电路

PLC 输出电路中没有保护，因此在外部电路中可以设置串联熔断器等保护装置，以防止负载短路造成 PLC 损坏。

（1）限位保护

对有些快速动作的机械设备不仅要有行程开关限位，还要有极限限位保护。限位保护可以由电子限位开关、机械限位开关和机械挡板等组成，如图 8-4 所示。


图 8-4　限位保护

（2）急停保护

急停保护是用来应付突发故障时，避免导致更大事故的措施。

（3）联锁保护

在硬件设计中要考虑到有些可移动设备的联锁问题。比如电动机的正转和反转接触器之间要有互锁保护。液压系统中双向电磁阀也要有互锁保护。互锁的作用是使得两个相反动作的设备不会同时动作。常用的办法是将两个设备的常闭触点相互串联，使用联锁开关等。

5. 系统接地设计

在实际的控制系统设计中，接地设计是抑制信号干扰、确保系统可靠工作的主要方法。所以，通常要单独设计 PLC 的接地系统。

在 PLC 组成的控制系统中，主要有以下几种地线：

1）数字地：这是各种数字量信号的零电位，也称为逻辑地。

2）模拟地：这是各种模拟量信号的零电位。

3）交流地：交流电源的地线，通常也是产生干扰和噪声的地。

4）直流地：直流电源的地线。

5）屏蔽地：这是为防止静电感应和磁场感应而设置的地。

（1）保护接地

保护接地指的是接大地，可采用不小于 10mm² 的保护铜导线接好配电板的保护地。相邻的控制柜也应良好接触并与地可靠连接，并尽可能短地与 UPS 电源、系统地线连接。同时要做好防雷保护接地，通常可采取总线电缆使用屏蔽电缆且屏蔽层两端接地，或模拟信号电缆采取两层屏蔽，外层屏蔽两端接地等措施。另外，为防止感应雷电进入系统，可采用浪涌吸收器。

（2）工作接地

工作接地包括信号回路接地和屏蔽接地。

1）在 PLC 控制系统中，非隔离信号需要有一个统一的信号参考点，并且进行信号回路接地。信号回路接地通常使用直流电源负极。

2）为防止静电感应和磁场感应而设置的屏蔽接地端子，应做屏蔽接地。

6. 使用环境

应合理布置 PLC 的使用环境，提高系统的抗干扰能力。具体采取的措施有：远离高压柜、高频设备、动力屏柜以及高压线或大电流动力装置；通信电缆和模拟信号电缆尽量不与其他的屏柜或设备共用电缆沟；PLC 柜内不用荧光灯等。另外，PLC 虽适合工业现场，但使用中也应尽量避免直接震动和冲击、阳光直射、油雾、雨淋等；不要在有腐蚀性气体、灰尘过多、发热体附近应用；避免导电性杂物进入控制器。

四、软件可靠性设计

1. 软件保护设计

（1）联锁保护

利用 PLC 的常闭触点对需要互锁的线圈进行联锁保护，如图 8-5 所示。

（2）双重保护

利用 PLC 的输入端增加限位输入信号，在程序中利用这类信号实现保护功能，如图 8-6 所示。

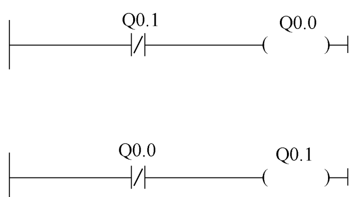


图 8-5　软件设计联锁保护

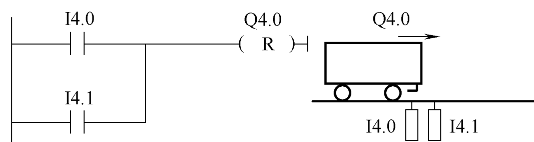


图 8-6　双重保护

（3）定时保护

根据物理知识，可以知道：距离 = 速度 × 时间，用时间代替距离。表示距离的物理示意图如图 8-7 所示。定时保护如图 8-8 所示。

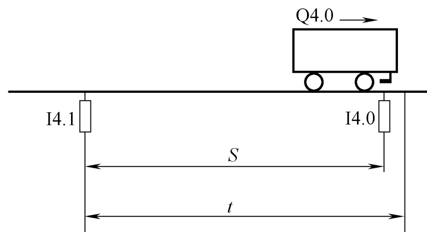


图 8-7　表示距离的物理示意图

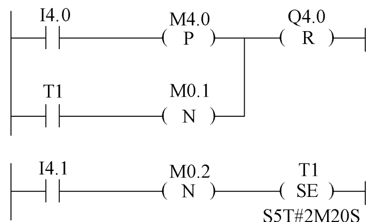


图 8-8　定时保护

（4） 自动复位保护

自动复位保护如图 8-9 所示。

2. 定时器的使用

PLC 的定时器可以用做确保 PLC 可靠、稳定、安全运行设计。定时器用做抗干扰设计，以解决限位开关的抖动干扰，如图 8-10 所示。

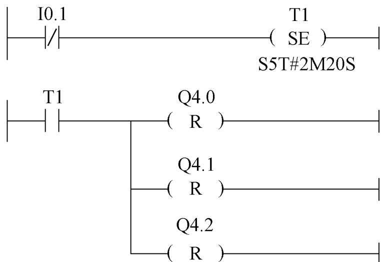


图 8-9 自动复位保护

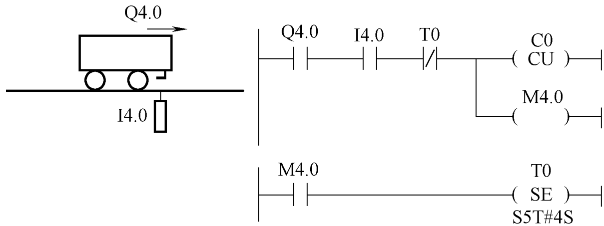


图 8-10 定时器的使用

3. 自检保护设计

可以充分利用 PLC 的自检功能，PLC 的多数功能模块都有自检信息，通过对这些自检信息的检查可以及时发现隐患并清除故障。也可以针对工程的特点自己编写诊断程序，排除故障。

4. 属性等级保护

S7-300 保护有读/写保护功能、写保护功能和不保护。保护等级可以在 CPU 属性中设定。程序的保护如图 8-11 所示。

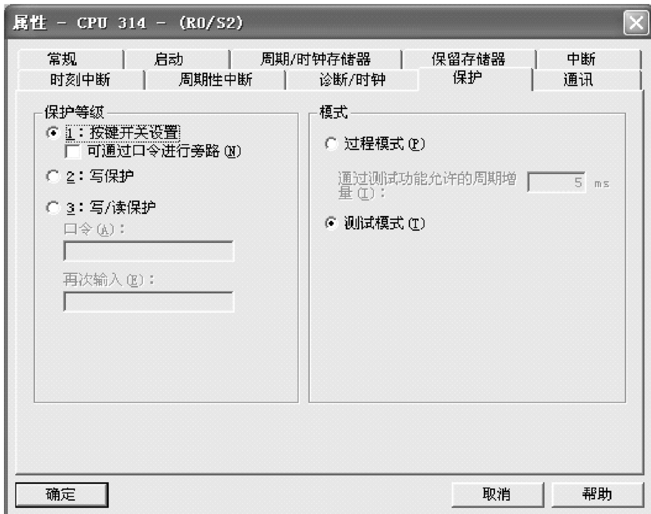


图 8-11 程序的保护

第七节 编程实例与工程应用

本章主要以一些简单编程和工程实例的控制系统为例讲解 PLC 的应用编程。

一、简单编程实例

例一、二分频器

二分频器是一种具有一个输入端和一个输出端的功能单元，输出频率为输入频率的一半。二分频时序图如图 8-12 所示，输入为 I0.0，输出为 Q4.0。

梯形图程序如图 8-13 所示。

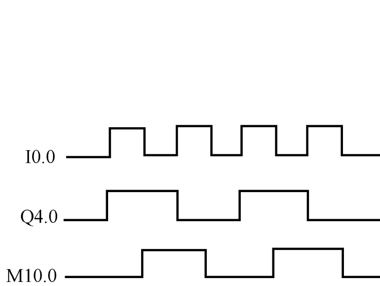


图 8-12　二分频时序图

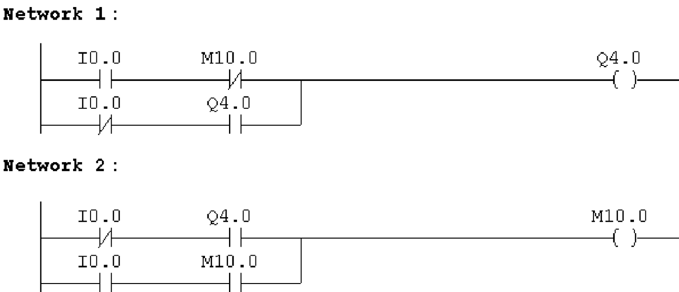


图 8-13　梯形图程序

语句表程序如图 8-14 所示。

例二、风机监控程序

某设备有三台风机，当设备处于运行状态时，如果风机至少有两台转动，则指示灯常亮；如果仅有一台风机转动，则指示灯以 0.5Hz 的频率闪烁；如果没有任何风机转动，则指示灯以 2Hz 的频率闪烁。当设备不运行时，指示灯不亮。

梯形图程序如图 8-15 所示。

语句表程序如图 8-16 所示。

输入位 I0.0、I0.1、I0.2 分别表示风机 1、2、3。存储位 M100.3 为 2Hz 的频率信号，M100.7 为 0.5Hz 的信号。风机转动状态指示灯由 Q4.0 控制。存储位 10.0 为 1 时用于表示至少有两台风机转动，M10.1 为 1 时表示没有风机转动。

```
Network 1:
      A     I     0.0
      AN    M    10.0
      O
      AN    I     0.0
      A     Q     4.0
      =     Q     4.0

Network 2:
      AN    I     0.0
      A     Q     4.0
      O
      A     I     0.0
      A     M    10.0
      =     M    10.0
```

图 8-14　语句表程序

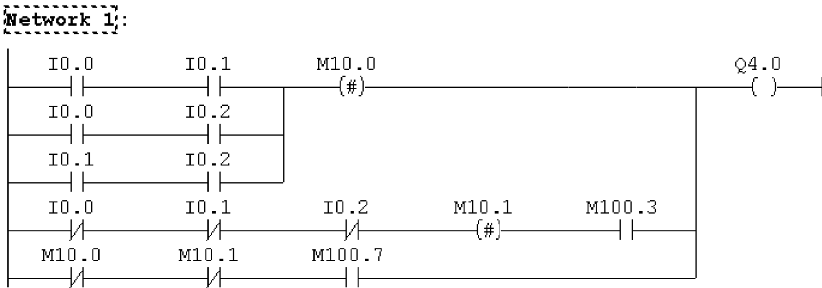


图 8-15　梯形图程序

例三、起动和自锁程序

程序功能：输入 X0 闭合时，输出 Y0 闭合且自锁。只有在 X1 闭合时，其动断触点打开，Y0 断开。其时序图如图 8-17 所示。

梯形图程序如图 8-18 所示。

语句表程序如图 8-19 所示。

Network 1:

```
A(
A     I     0.0
A     I     0.1
O
A     I     0.0
A     I     0.2
O
A     I     0.1
A     I     0.2
)
=     M     10.0
A     M     10.0
O(
AN    I     0.0
AN    I     0.1
AN    I     0.2
=     M     10.1
A     M     10.1
A     M     100.3
)
O
AN    M     10.0
AN    M     10.1
A     M     100.7
=     Q     4.0
```

图 8-16　语句表程序

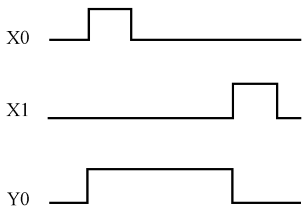


图 8-17　时序图

Network 1:

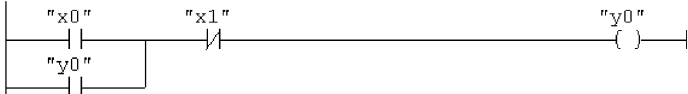


图 8-18　梯形图程序

Network 1:

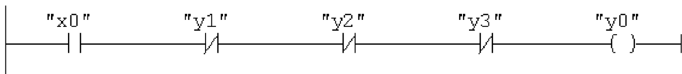
```
A(
O     "x0"
O     "y0"
)
AN    "x1"
=     "y0"
```

图 8-19　语句表程序

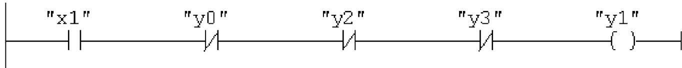
例四、优先程序

优先程序执行时，能在多个输入信号中仅接收最先一个输入的信号做出反应，其后的输入信号不接收。此原则常用于抢答器中。梯形图程序如图 8-20 所示。

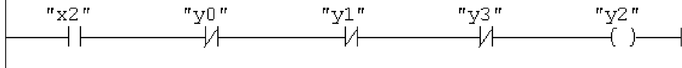
Network 1:



Network 2:



Network 3:



Network 4:

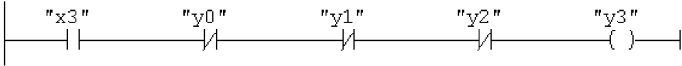


图 8-20　梯形图程序

例五、灯泡控制程序

一盏灯泡由一个按钮来控制，已知第一次按下按钮，灯泡亮，第二次按下按钮，灯泡灭。

1. PLC 接线图（见图 8-21）

2. 定义符号地址（见表 8-2）

3. 梯形图程序（见图 8-22）

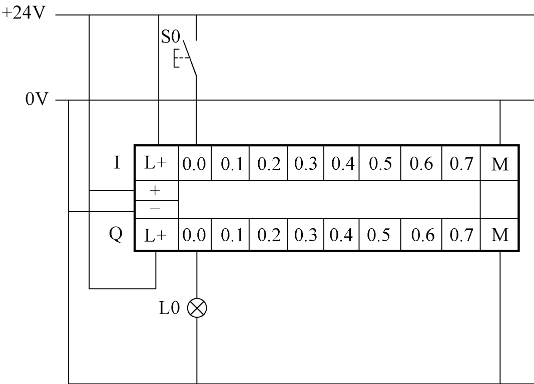


图 8-21　PLC 接线图

表 8-2　定义符号地址

符号地址	绝对地址	类据类型	说明
S0	I0.0	BOOL	按钮
L0	Q0.0	BOOL	灯泡
M0	M0.0	BOOL	标志位

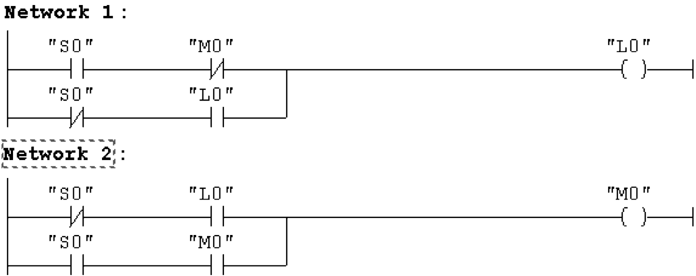


图 8-22　梯形图程序

例六、双作用气缸连续往复运动控制

按起动按钮双作用气缸连续往复运动，按停止按钮，停止运动。

1. 气控回路（见图 8-23）

2. PLC 接线（见图 8-24）

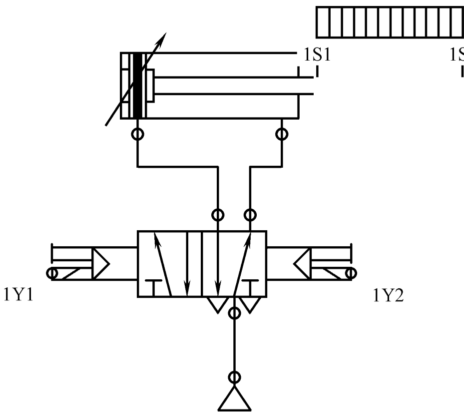


图 8-23　气控回路

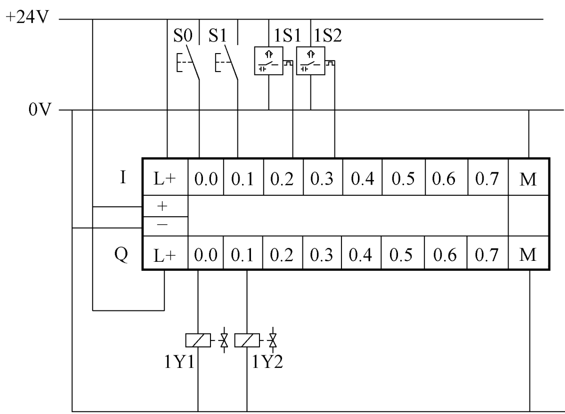


图 8-24　PLC 接线

3. 定义符号地址（见表 8-3）

表 8-3　定义符号地址

符号地址	绝对地址	类据类型	说明	符号地址	绝对地址	类据类型	说明
S0	I0.0	BOOL	起动按钮	1Y1	Q0.0	BOOL	换向阀电磁线圈
S1	I0.1	BOOL	停止按钮	1Y2	Q0.1	BOOL	换向阀电磁线圈
1S1	I0.2	BOOL	位置传感器	M0	M0.0	BOOL	起动线圈
1S2	I0.3	BOOL	位置传感器				

4. 梯形图程序（见图 8-25）

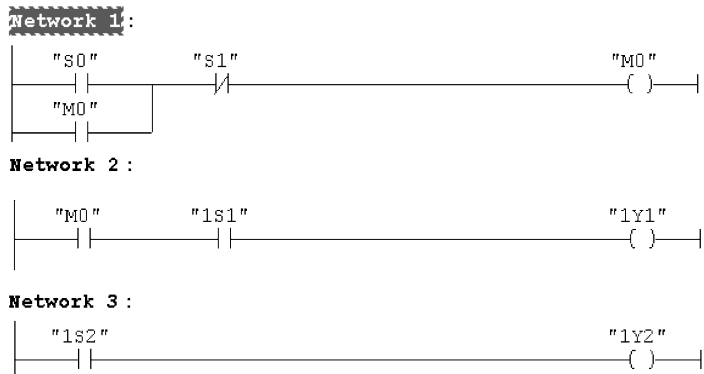


图 8-25　梯形图程序

例七、控制传送带

一个由电气起动的传送带，在传送带的起点有两个按钮：用于 START 的 S1 和 STOP 的 S2。在传送带的尾部也有两个按钮：用于 START 的 S3 和 STOP 的 S4。可以从任何一端起动或停止传送带。另外，当传送带上的物件到达末端时，传感器 S5 使传送带停机。

1. PLC 接线（见图 8-26）

2. 定义符号地址（见表 8-4）

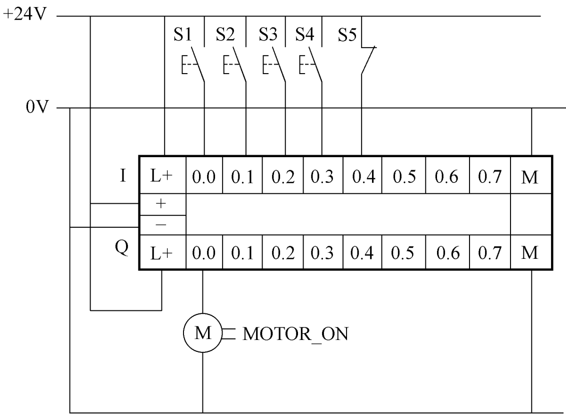


图 8-26　PLC 接线

表 8-4　定义符号地址

符号地址	绝对地址	类据类型	说明
S1	I0.0	BOOL	起点起动按钮
S2	I0.1	BOOL	起点停机按钮
S3	I0.2	BOOL	尾部起动按钮
S4	I0.3	BOOL	尾部停机按钮
S5	I0.4	BOOL	末端传感器
MOTOR_ON	Q0.0	BOOL	电动机

3. 梯形图程序（见图 8-27）

例八、双缸顺序动作控制程序

设计程序，使两个气缸顺序动作，其顺序为：A1B1B0A0。

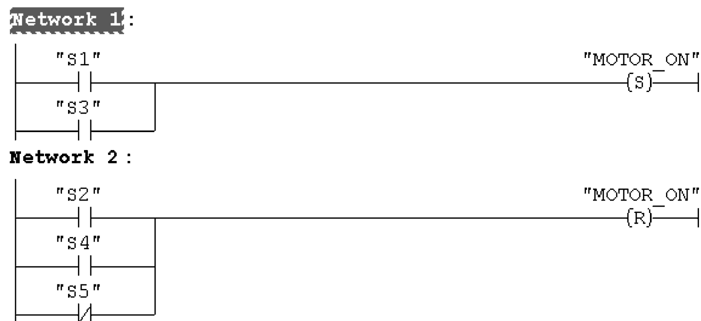


图 8-27　梯形图程序

1. 气控回路（见图 8-28）

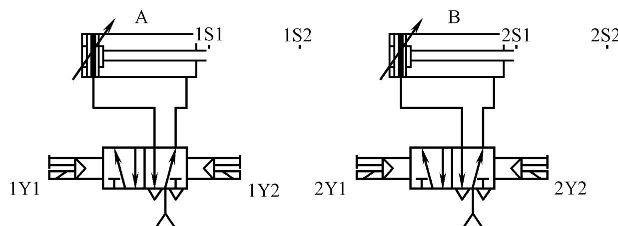


图 8-28　气控回路

2. 位移-步骤图（见图 8-29）

3. I 型障碍信号分析（见图 8-30）

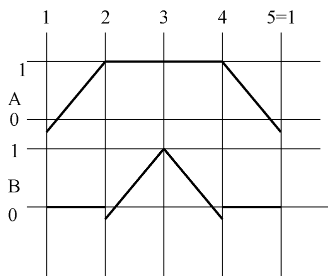


图 8-29　位移-步骤图

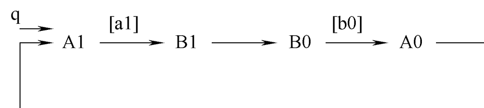


图 8-30　I 型障碍信号分析

4. PLC 接线（见图 8-31）

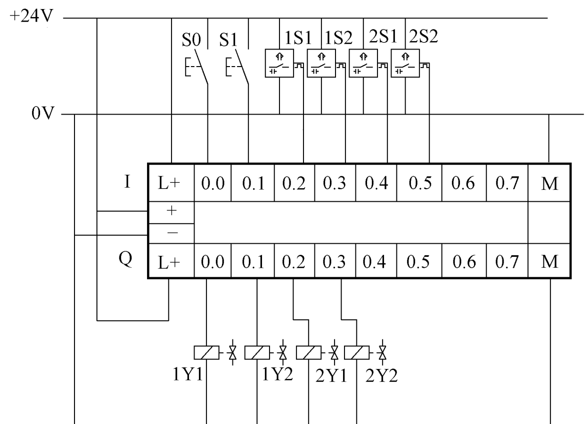


图 8-31　PLC 接线

5. 定义符号地址（见表 8-5）

表 8-5 定义符号地址

序号	符号地址	绝对地址	数据类型	序号	符号地址	绝对地址	数据类型
1	S0	I0.0	BOOL	9	2Y1	Q0.2	BOOL
2	S1	I0.1	BOOL	10	2Y2	Q0.3	BOOL
3	1S1	I0.2	BOOL	11	STEP0	M0.0	BOOL
4	1S2	I0.3	BOOL	12	STEP1	M0.1	BOOL
5	2S1	I0.4	BOOL	13	STEP2	M0.2	BOOL
6	2S2	I0.5	BOOL	14	STEP3	M0.3	BOOL
7	1Y1	Q0.0	BOOL	15	STEP4	M0.4	BOOL
8	1Y2	Q0.1	BOOL				

6. 梯形图程序（见图 8-32）

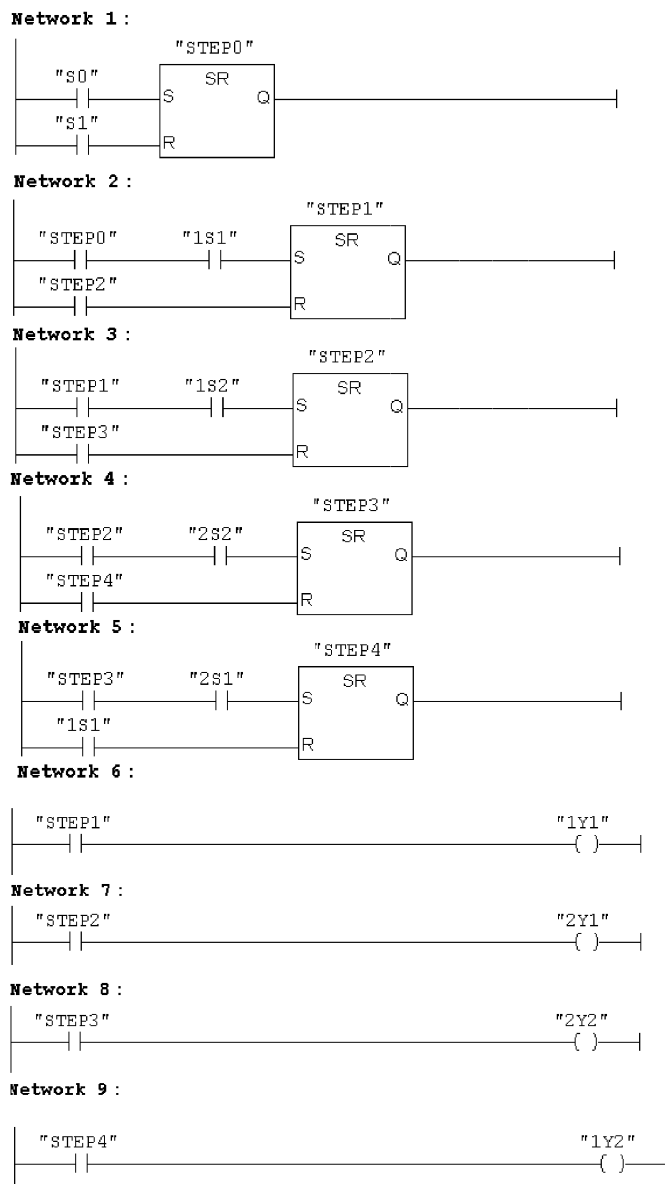


图 8-32 梯形图程序

例九、检测传送带的方向

装备有两个光电传感器（PEB1 和 PEB2）的传送带如图 8-33 所示，该设计能够检测传送带上物件的运动方向，并通过左右两端的指示灯（LEFT 灯和 RIGHT 灯）显示。

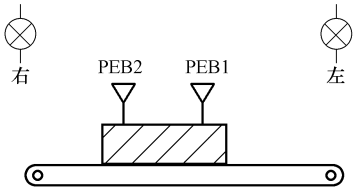


图 8-33　装备有两个光电传感器的传送带

1. PLC 接线（见图 8-34）

2. 定义符号地址（见表 8-6）

3. 梯形图程序（见图 8-35）

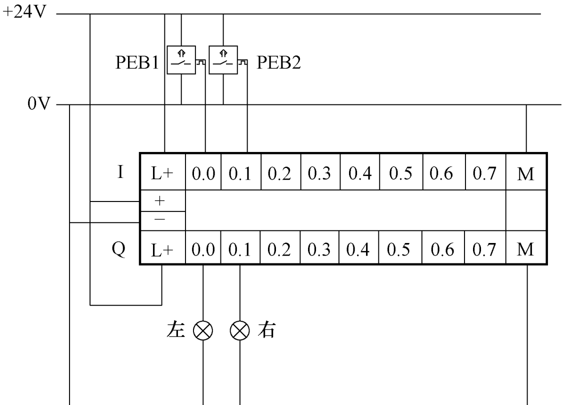


图 8-34　PLC 接线

表 8-6　定义符号地址

序号	符号地址	绝对地址	数据类型
1	PEB1	I0.0	BOOL
2	PEB2	I0.1	BOOL
3	LEFT	Q0.0	BOOL
4	RIGHT	Q0.1	BOOL
5	PMB1	M0.0	BOOL
6	PMB2	M0.1	BOOL

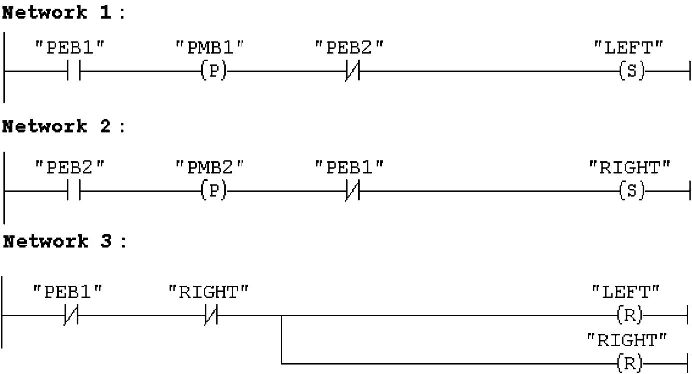


图 8-35　梯形图程序

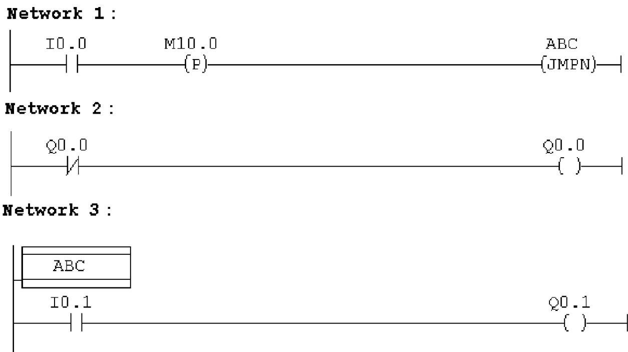


图 8-36　梯形图程序 1

例十、二分频器

二分频器是一种具有一个输入端和一个输出端的功能单元，输出频率为输入频率的一半。二分频时序图如图 8-12 所示，输入为 I0.0，输出为 Q4.0。

分析二分频时序图可以看到，输入每有一个正跳沿，输出便反转一次。据此，可用跳变沿检测指令实现分频功能。

梯形图程序 1 如图 8-36 所示。

梯形图程序 2 如图 8-37 所示。

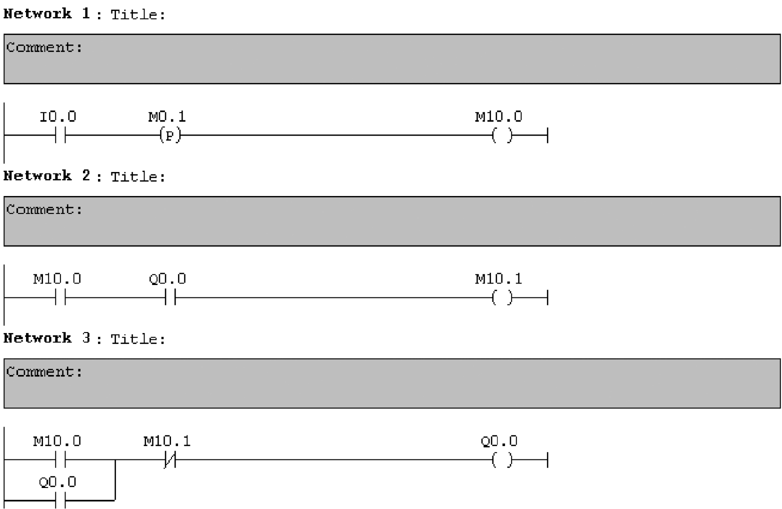


图 8-37　梯形图程序 2

例十一、传送带定位控制

一台电动机带动一个传送带运动，要求移动传送带向前或向后到达某一确定的位置，其结构示意图如图 8-38 所示，为了正确定位该传送带，有时需要按下向后（REV）或向前（FWD）按钮进行手动调整。

梯形图程序如图 8-39 所示。

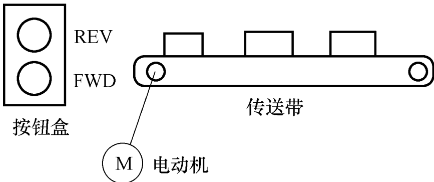


图 8-38　结构示意图

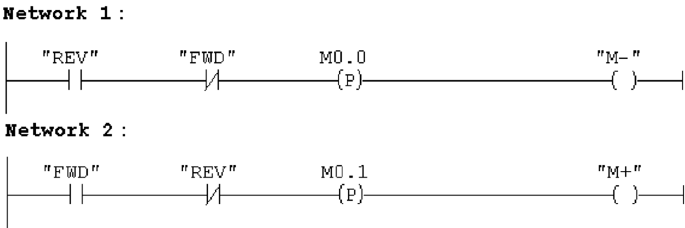


图 8-39　梯形图程序

一旦有按钮按下，立即驱动输出，电动机运转一个扫描周期。这也意味着按钮时间长短与电动机驱动的时间没有关系。

例十二、循环控制指示灯

第一次按按钮指示灯亮，第二次按按钮指示灯闪亮，第三次按按钮指示灯灭，如此循环，试编写其 PLC 控制的梯形图程序。

梯形图程序如图 8-40 所示。

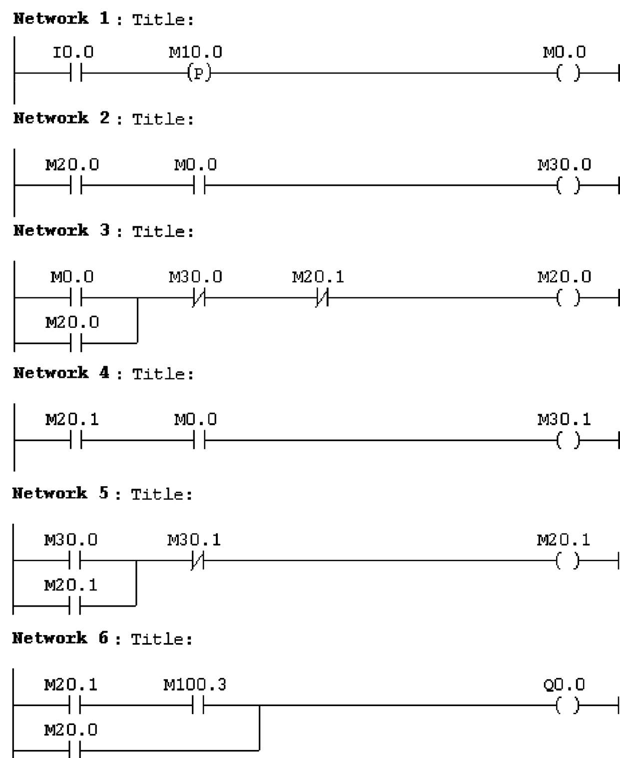


图 8-40　梯形图程序

例十三、脉冲发生器

用定时器可构成脉冲发生器，这里用了两个定时器产生频率占空比均可设置的脉冲信号。脉冲发生器时序图如图 8-41 所示，当输入 I0.0 为 1 时，输出 Q0.0 为 1 或 0 交替进行，脉冲信号的周期为 3s，脉冲宽度为 1s。

梯形图程序如图 8-42 所示。

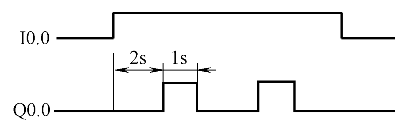


图 8-41　脉冲发生器时序图

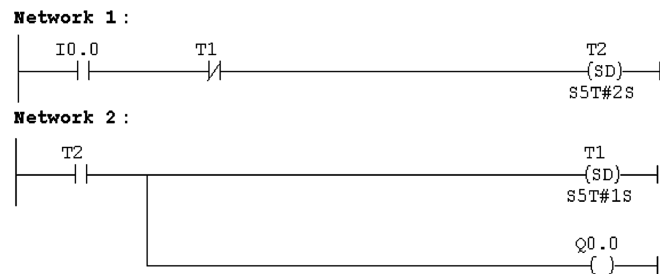


图 8-42　梯形图程序

例十四、频率监测器

频率监测器用于监测脉冲信号的频率，若其低于下限，则指示灯亮，“确认”按键能使指示灯复位。为此，使用了一个扩展脉冲定时器，每当频率信号有一个上升沿就起动一次定时器。如果超过了定时时间没有起动定时器，则表明两个脉冲之间的时间间隔太长，即频率太低了。频率监测器时序图如图 8-43 所示。

梯形图程序如图 8-44 所示。

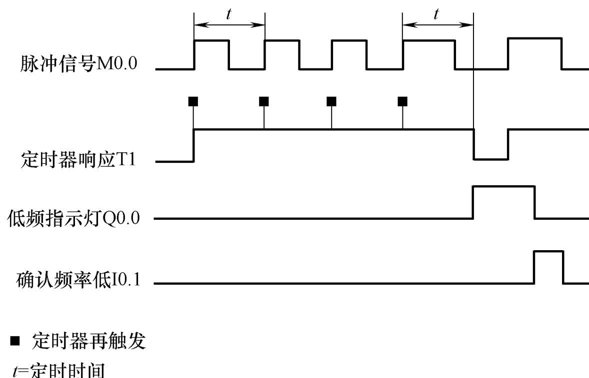


图 8-43 频率监测器时序图

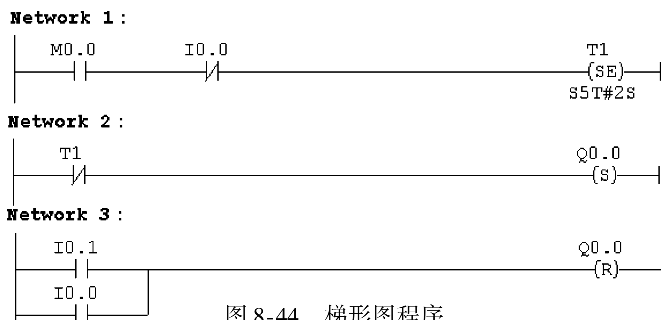


图 8-44 梯形图程序

例十五、顺序循环执行程序

当 X0 接通，灯 Y0 亮；经 5s 后，灯 Y0 灭，灯 Y1 亮；经 5s 后，灯 Y1 灭，灯 Y2 亮，再过 5s 后，灯 Y2 灭，灯 Y0 亮，如此顺序循环，其时序图如图 8-45 所示。

梯形图程序如图 8-46 所示。

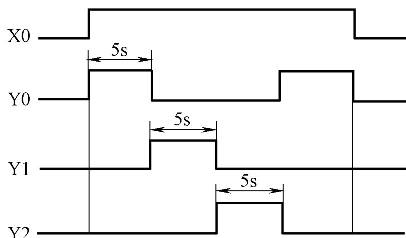


图 8-45 顺序循环执行程序时序图

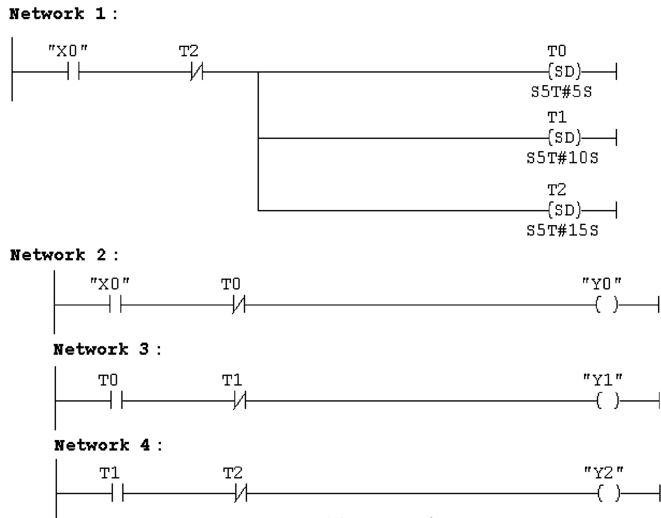


图 8-46 梯形图程序

例十六、电动机顺序起动控制程序

有三台电动机 M1、M2、M3，按下起动按钮后 M1 起动，延时 5s 后 M2 起动，再延时 16s 后 M3 起动。

1. PLC 接线（见图 8-47）

2. 定义符号地址（见表 8-7）

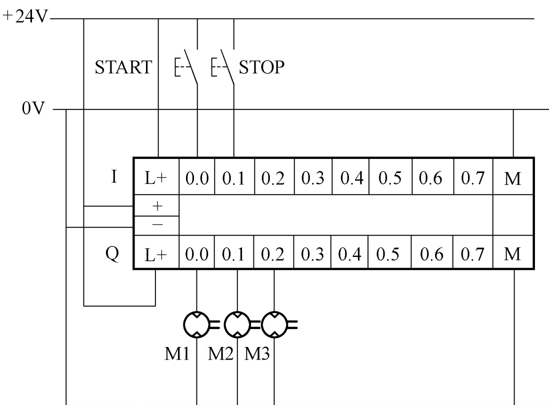


图 8-47　PLC 接线

表 8-7　定义符号地址

序号	符号地址	绝对地址	数据类型
1	START	I0.0	BOOL
2	STOP	I0.1	BOOL
3	M1	Q0.0	BOOL
4	M2	Q0.1	BOOL
5	M3	Q0.2	BOOL

3. 梯形图程序（见图 8-48）

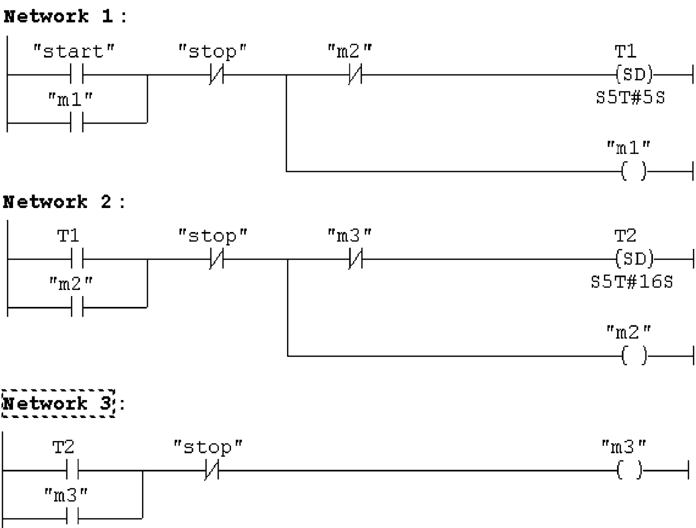


图 8-48　梯形图程序

例十七、分段传送带的电动机控制程序

为了节省能源的损耗，可使用 PLC 来起动和停止分段传送带的驱动电动机，使那些只载有物体的传送带运转，没有载物的传送带停止运行。金属板正在传送带上输送，其位置由相应的传感器检测。传感器安放在两段传送带相邻

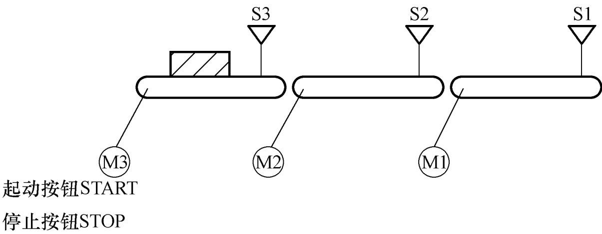


图 8-49　运动过程示意图

近的地方，一旦金属板进入传感器的检测范围，PLC 便发出相应的输出信号，使后一段传送带的电动机投入工作；当金属板被送出检测范围时，PLC 内部定时器立即开始计时，在达到预定的延时时间后，前一段传送带电动机便停止运行。运动过程示意图如图 8-49 所示。

1. PLC 接线（见图 8-50）

2. 定义符号地址（见表 8-8）

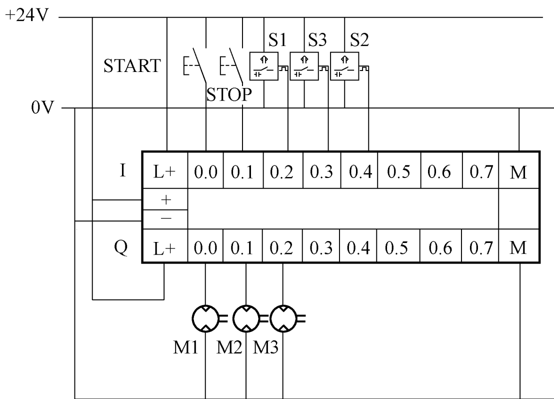


图 8-50　PLC 接线

表 8-8　定义符号地址

序号	符号地址	绝对地址	数据类型
1	M1	Q0.0	BOOL
2	M2	Q0.1	BOOL
3	M3	Q0.2	BOOL
4	START	I0.0	BOOL
5	STOP	I0.1	BOOL
6	S1	I0.2	BOOL
7	S2	I0.3	BOOL
8	S3	I0.4	BOOL

3. 梯形图程序（见图 8-51）

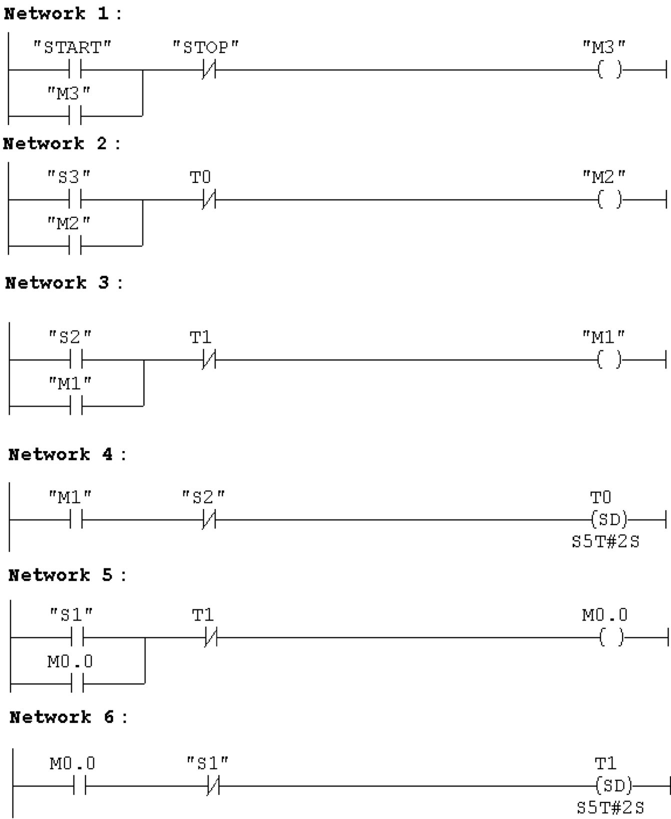


图 8-51　梯形图程序

例十八、三台电动机顺起逆停

如图 8-52a 所示 A、B、C 三条传送带，分别受 M0、M1、M2 三台电动机拖动；图 8-52b 所示是此三条传送带运转的时序图。编写一个用 PLC 控制它们运转的梯形图程序。要求它们按 A—B—C 顺序起动，而后按 C—B—A 的顺序停止。

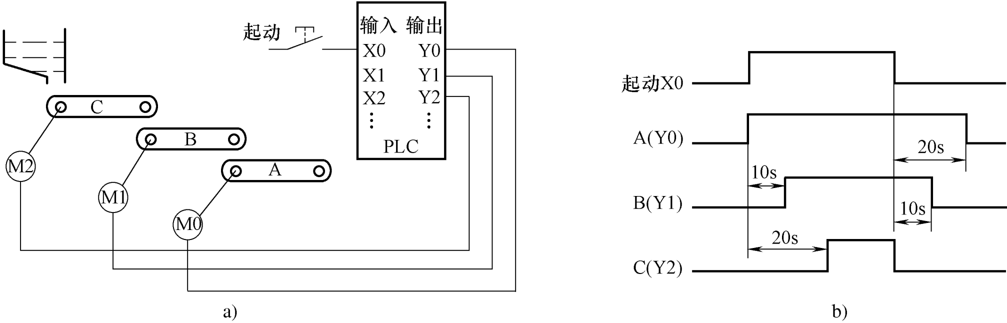


图 8-52　三台电动机顺起逆停

a）运动示意图　b）运转时序图

梯形图程序如图 8-53 所示。

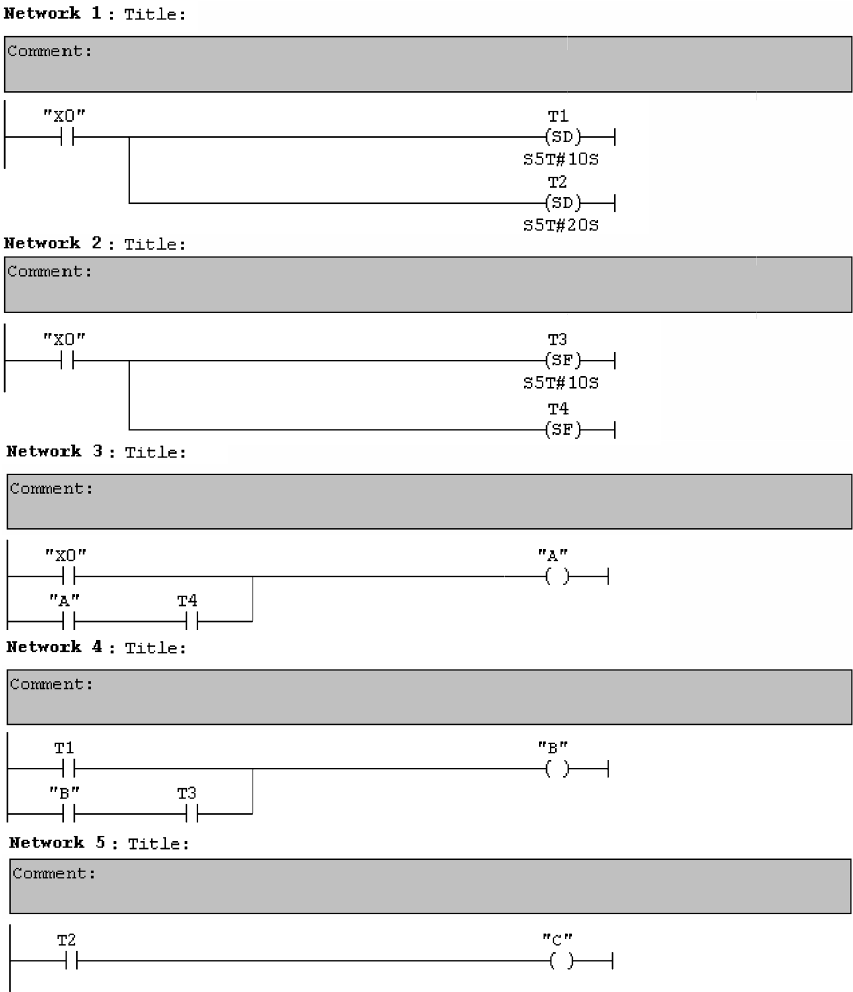


图 8-53　梯形图程序

例十九、十字路口的交通指挥信号灯

交通指挥信号灯布置如图 8-54 所示。

1. 控制要求

1）信号灯系统由一个起动开关控制，当起动开关接通时，该信号灯系统开始工作，当起动开关关断时，所有信号灯都熄灭。

2）南北绿灯和东西绿灯不能同时亮。如果同时亮应关闭信号灯系统，并立刻报警。

3）南北红灯亮维持 25s。在南北红灯亮的同时东西绿灯也亮，并维持 20s。到 20s 时，东西绿灯闪亮，闪亮 3s 后熄灭，此时，东西黄灯亮，并维持 2s。到 2s 时，东西黄灯熄灭，东西红灯亮。同时，南北红灯熄灭，南北绿灯亮。

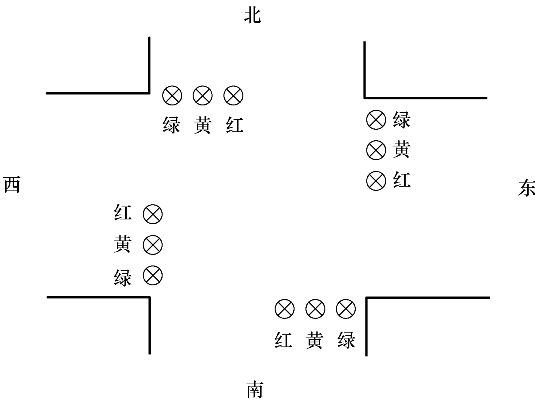


图 8-54　交通指挥信号灯布置

4）东西红灯亮维持 30s。南北绿灯亮维持 25s，然后闪亮 3s 后熄灭。同时南北黄灯亮，维持 2s 后熄灭，这时南北红灯亮，东西绿灯亮。

5）以上南北、东西信号灯周而复始地交替工作状态，指挥着十字路口的交通。其时序图如图 8-55 所示。

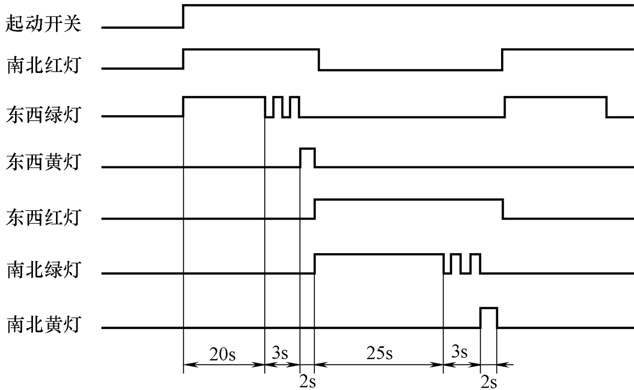


图 8-55　时序图

2. PLC 接线（见图 8-56）

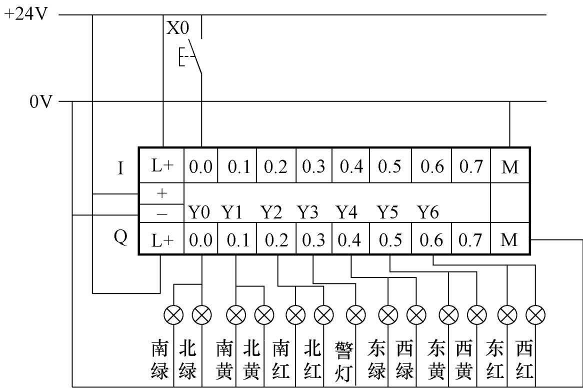


图 8-56　PLC 接线

3. 定义符号地址（见表 8-9）

表 8-9　定义符号地址

序号	符号地址	绝对地址	数据类型
1	X0	I0.0	BOOL
2	Y0	Q0.0	BOOL
3	Y1	Q0.1	BOOL
4	Y2	Q0.2	BOOL
5	Y3	Q0.3	BOOL
6	Y4	Q0.4	BOOL
7	Y5	Q0.5	BOOL
8	Y6	Q0.6	BOOL

4. 梯形图程序（见图 8-57）

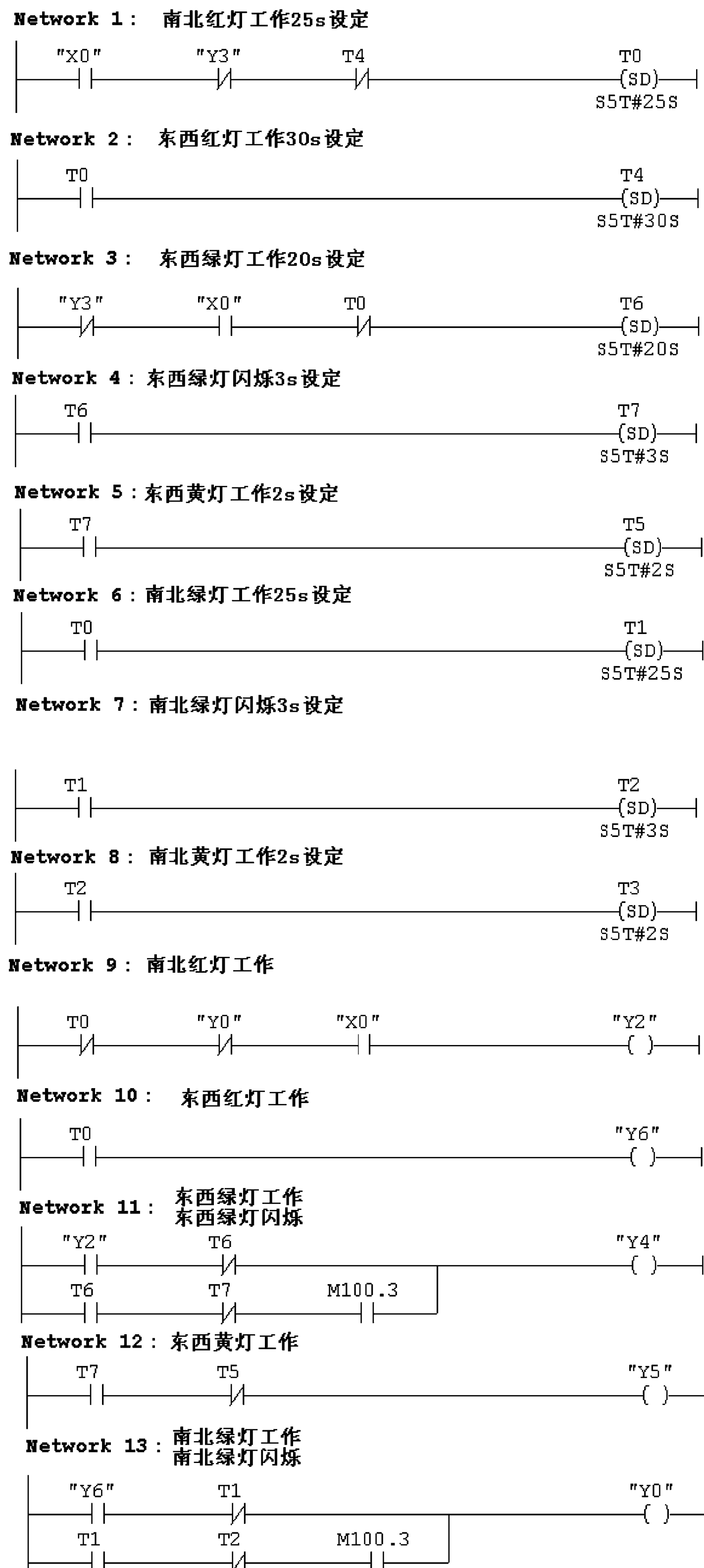


图 8-57 梯形图程序

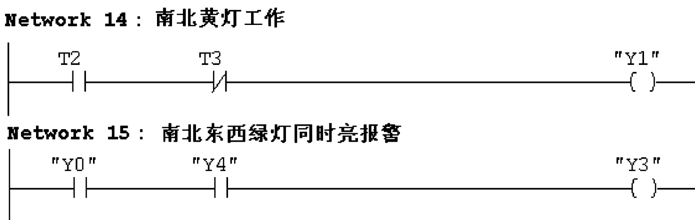


图 8-57 梯形图程序（续）

例二十、PLC 控制的气缸延时控制回路

当气缸的活塞伸出到位停留 5s 后再返回，退回到位停留 3s 后再伸出，如此往复运动。

1. 气控回路（见图 8-58）

2. PLC 接线（见图 8-59）

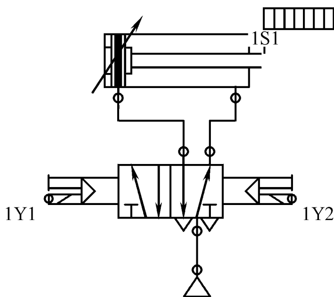


图 8-58 气控回路

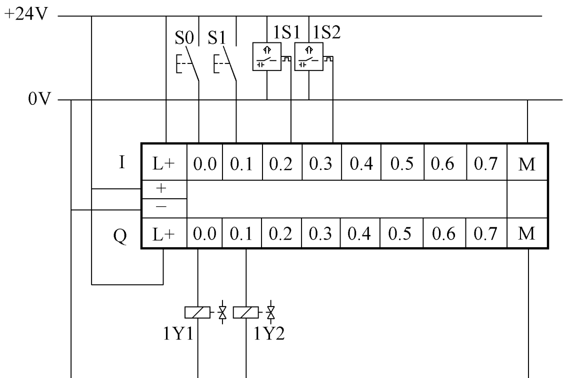


图 8-59 PLC 接线

3. 定义符号地址（见表 8-10）

表 8-10 定义符号地址

符号地址	绝对地址	类据类型	说明	符号地址	绝对地址	类据类型	说明
S0	I0. 0	BOOL	起动按钮	1Y1	Q0. 0	BOOL	换向阀电磁线圈
S1	I0. 1	BOOL	停止按钮	1Y2	Q0. 1	BOOL	换向阀电磁线圈
1S1	I0. 2	BOOL	位置传感器	M0	M0. 0	BOOL	起动线圈
1S2	I0. 3	BOOL	位置传感器				

4. 梯形图程序（见图 8-60）

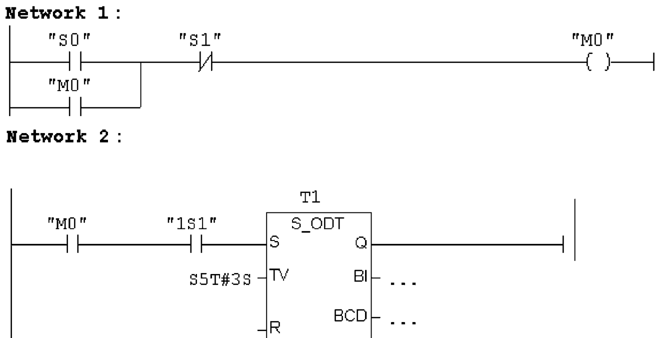


图 8-60 梯形图程序

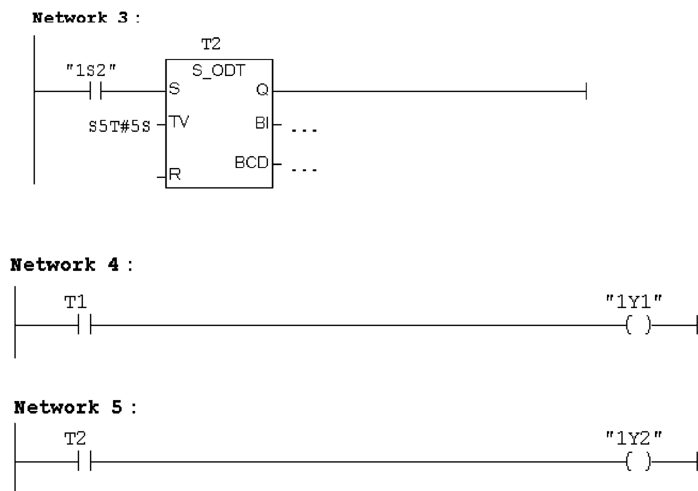


图 8-60　梯形图程序（续）

例二十一、多种液体自动混合装置的 PLC 控制

如图 8-61 所示为三种液体混合装置，SQ1、SQ2、SQ3 和 SQ4 为液面传感器，液面淹没时接通，液体 A、B、C 与混合液阀由电磁阀 YV1、YV2、YV3、YV4 控制，M 为搅匀电动机，其控制要求如下。

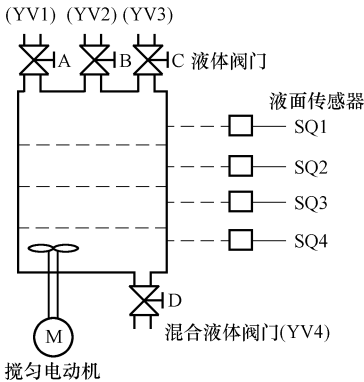


图 8-61　三种液体混合装置

1. 初始状态

装置投入运行时，液体 A、B、C 阀门关闭，混合液阀门打开 20s 将容器放空后关闭。

2. 起动操作

按下起动按钮 SB1，装置开始按下列给定规律运转：

1）液体 A 阀门打开，液体 A 流入容器。当液面达到 SQ3 时，SQ3 接通，关闭液体 A 阀门，打开液体 B 阀门。

2）当液面达到 SQ2 时，关闭液体 B 阀门，打开液体 C 阀门。

3）当液面达到 SQ1 时，关闭液体 C 阀门，搅匀电动机开始搅拌。

4）搅匀电动机工作 1min 后停止搅动，混合液体阀门打开，开始放出混合液体。

5）当液面下降到 SQ4 时，SQ4 由接通变为断开，再过 20s 后，容器放空，混合液阀门关闭，开始下一周期。

3. 停止操作

按下停止按钮 SB2 后，要将当前的混合操作处理完毕后，才停止操作（停在初始状态）

4. 参考程序（见图 8-62）

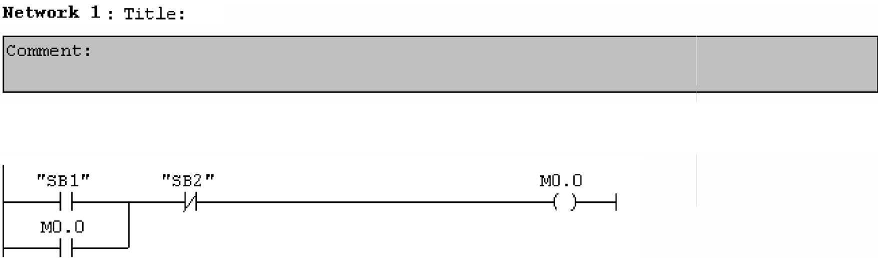
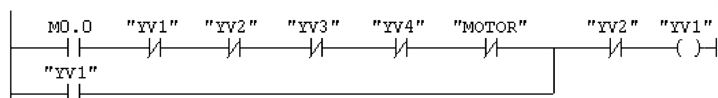


图 8-62　参考程序

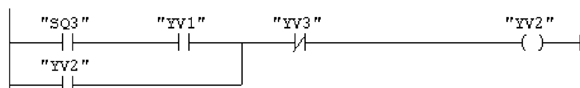
Network 2 : Title:

Comment:



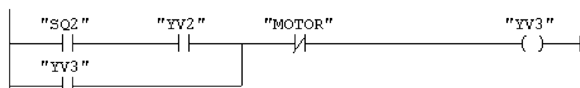
Network 3 : Title:

Comment:



Network 4 : Title:

Comment:



Network 5 : Title:

Comment:

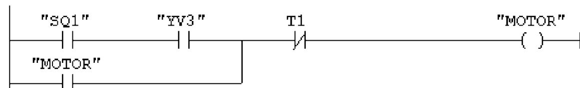


图 8-62 参考程序（续）

例二十二、计数器扩展为定时器

当定时器不够用时，可以用计数器扩展为定时器。程序中使用了 CPU 的时钟存储器，设置 MB100 为时钟存储器，则 M100.0 的变化周期为 0.1s。梯形图程序如图 8-63 所示。

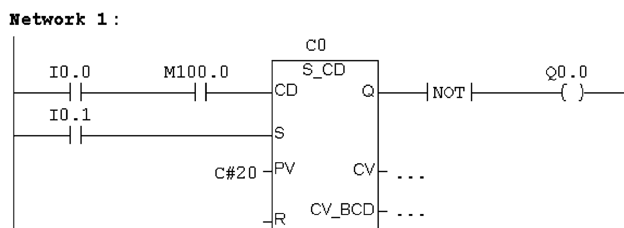


图 8-63 梯形图程序

在程序中，如果 I0.1 的正跳沿为减计数器 C0 置数，若 I0.0 为 1，则 C0 每 0.1s 减 1。当 C0 减到 0 后，输出 Q0.0 为 1，I0.1 的又一个正跳沿使 C0 置数并使输出为 0。这样，在 I0.0 为 1 后 2s（20 * 0.1s = 2s），Q0.0 为 1，I0.1 的正跳沿使 Q0.0 复位。

例二十三、长时间延时程序

采用定时器和计数器可以组成长时间延时程序，其时序图如图 8-64 所示。

梯形图程序如图 8-65 所示。

当输入 I0.0 接通时，定时器 T0 经过 10s 时间延时后，其动合触点 T0 闭合，计数器 C0 开始递减运算，与此同时 T0 的动断触点是断开的，造成 T0 线圈断电，使 T0 的动合触点断开，C0 仅计数一次，而后 T0 线圈又接通，如此循环。当 C0 经过 10s * 10 = 100s 时间后，计数器 C0 输出为 0，输出 Q0.0 接通，具有长时间延时的功能。

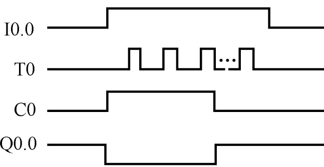


图 8-64　时序图

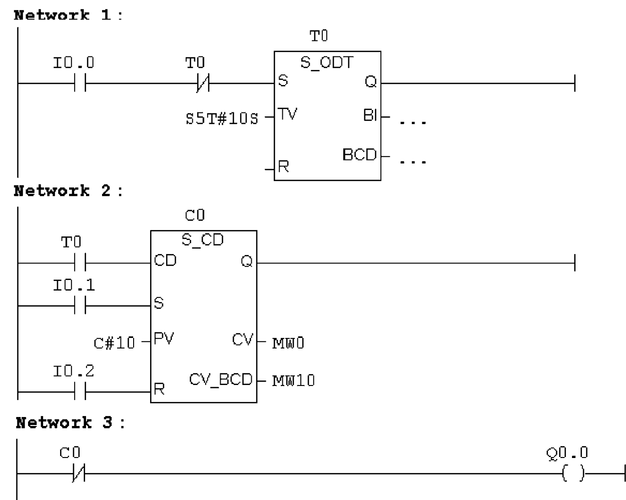


图 8-65　梯形图程序

例二十四、货仓区的控制

货仓区的控制如图 8-66 所示，装有两台传送带的系统，在两台传送带之间有一个仓库区。传送带 1 将包裹运送至临时仓库区。传送带 1 靠近仓库区一端安装的光电传感器确定已有多少包裹运送至仓库区。传送带 2 将临时仓库区中的包裹运送至装货场，在这里货物由卡车运送至顾客。传送带 2 靠近仓库区一端安装的光电传感器确定已有多少包裹从仓库区运送至装货场。

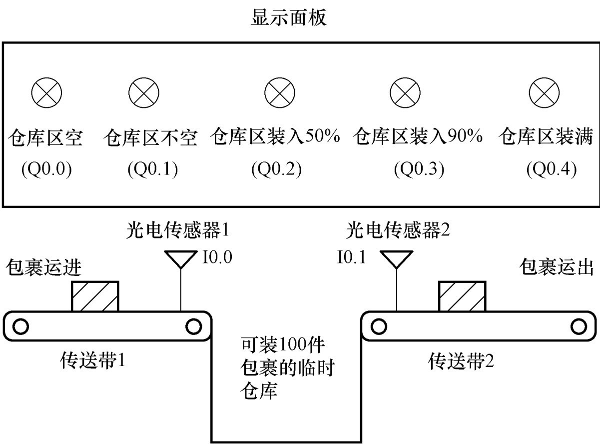


图 8-66　货仓区的控制

梯形图程序如图 8-67 所示。

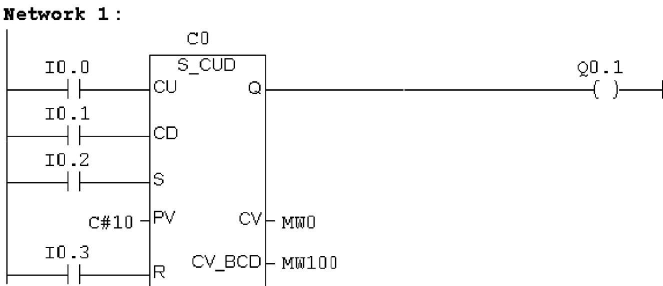


图 8-67　梯形图程序

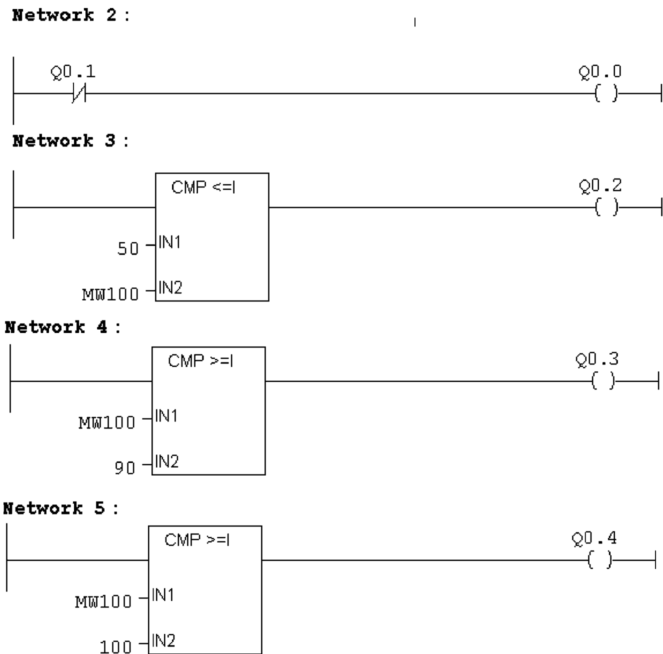


图 8-67　梯形图程序（续）

例二十五、气缸运动计数控制

要求：气缸连续往复运动 20 次便自动停止。

1. 气控回路（见图 8-68）

2. PLC 接线（见图 8-69）

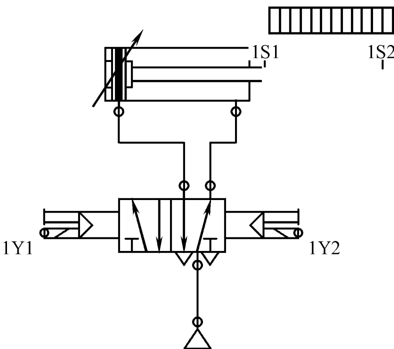


图 8-68　气控回路

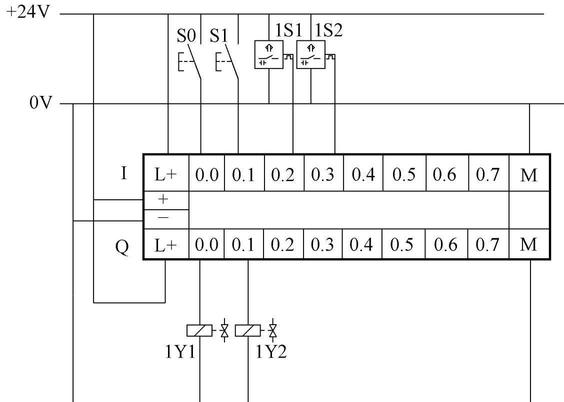


图 8-69　PLC 接线

3. 定义符号地址（见表 8-11）

表 8-11　定义符号地址

符号地址	绝对地址	数据类型	说明	符号地址	绝对地址	数据类型	说明
S0	I0. 0	BOOL	起动按钮	1Y1	Q0. 0	BOOL	换向阀电磁线圈
S1	I0. 1	BOOL	停止按钮	1Y2	Q0. 1	BOOL	换向阀电磁线圈
1S1	I0. 2	BOOL	位置传感器	M0	M0. 0	BOOL	起动线圈
1S2	I0. 3	BOOL	位置传感器				

4. 梯形图程序（见图 8-70）

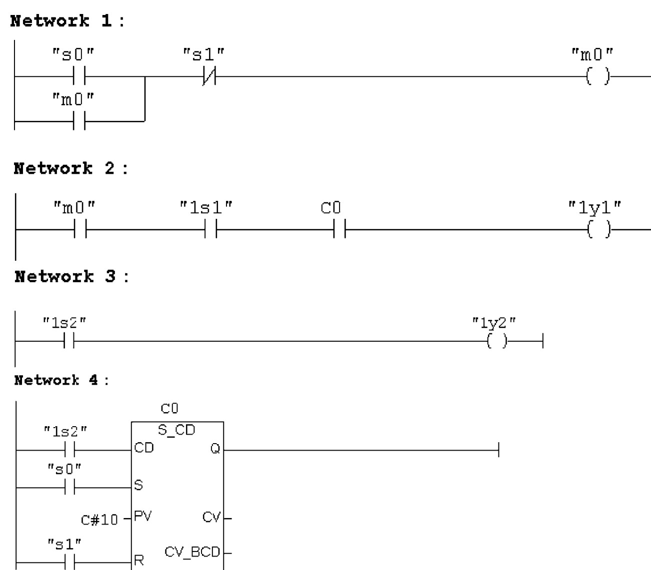


图 8-70　梯形图程序

例二十六、循环控制 3 个灯

当 X0 接通，灯 Y0 亮；经 5s 后，灯 Y0 灭，灯 Y1 亮；经 5s 后，灯 Y1 灭，灯 Y2 亮，再过 5s 后，灯 Y2 灭，灯 Y0 亮，如此顺序循环 10 次后自动停止。

梯形图程序如图 8-71 所示。

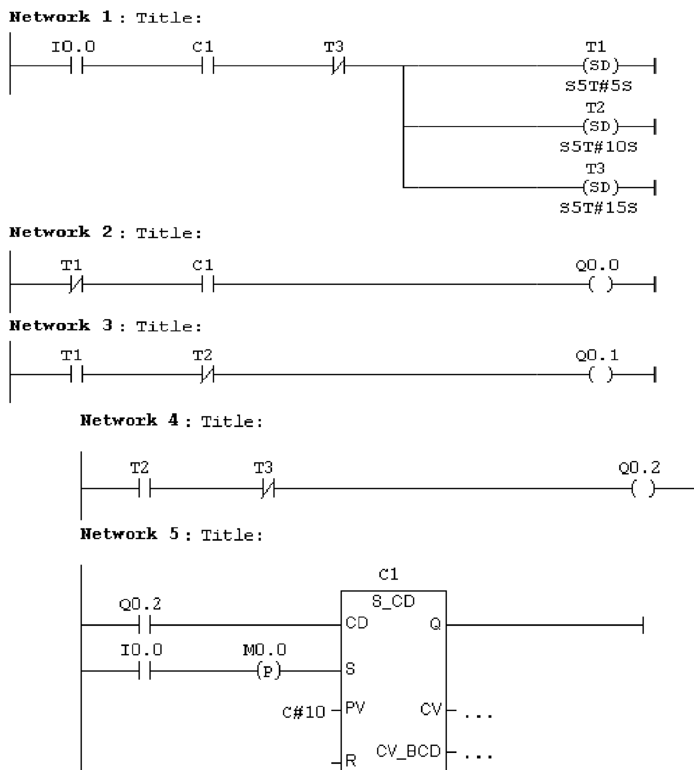


图 8-71　梯形图程序

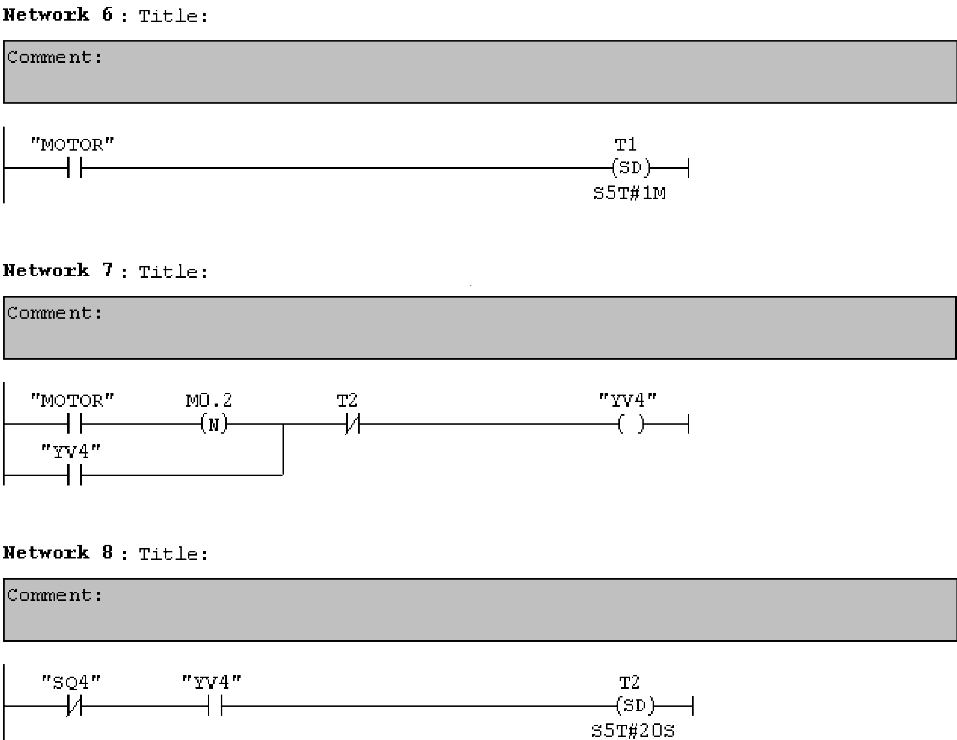


图 8-71　梯形图程序（续）

例二十七、三层楼电梯控制程序

如图 8-72 所示是三层楼电梯示意图。电梯的上升，下降由一台电动机控制；正转时电梯上升，反转时电梯下降。各层设一个呼叫开关（SB1、SB2、SB3）、一个呼叫指示灯（H1、H2、H3）、一个到位行程开关（ST1、ST2、ST3）。

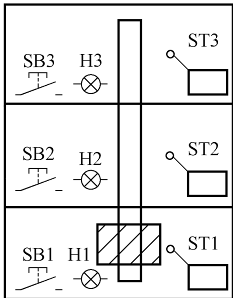


图 8-72　三层楼电梯示意图

控制要求：

1）各层的呼叫开关为按钮式开关，SB1、SB2 及 SB3 均为瞬间接通有效（瞬间接通的即放开仍有效）。

2）电梯箱体上升途中只响应上升呼叫，下降途中只响应下降呼叫，任何反方向呼叫均无效，简称为不可逆响应。具体动作要求，见表 8-12。

3）各楼层间有效运行时间应小于 10s，否则认为有故障，自动令电动机停转。

表 8-12　动作要求

序号	输　入		输　出	
	原停层	呼叫层	运行方向	运　行　结　果
1	1	3	升	上升到 3 层停，这期间经过 2 层时不停
2	2	3	升	上升到 3 层停
3	3	3	停	呼叫无效
4	1	2	升	上升到 2 层停
5	2	2	停	呼叫无效
6	3	2	降	下降到 2 层停
7	1	1	停	呼叫无效

（续）

序号	输　　入		输　　出	
	原停层	呼叫层	运行方向	运　行　结　果
8	2	1	降	下降到 1 层停
9	3	1	降	下降到 1 层停，这期间经过 2 层时不停
10	1	2、3	升	先升到 2 层暂停 2s 后，再升到 3 层停
11	2	1、3	降	下降到 1 层停
12	2	3、1	升	上升到 3 层停
13	3	2、1	降	先降到 2 层暂停 2s 后，再降到 1 层停
14	任意	任意	任意	楼层间运行时间必须小于 10s，否则停

梯形图程序如图 8-73 所示。

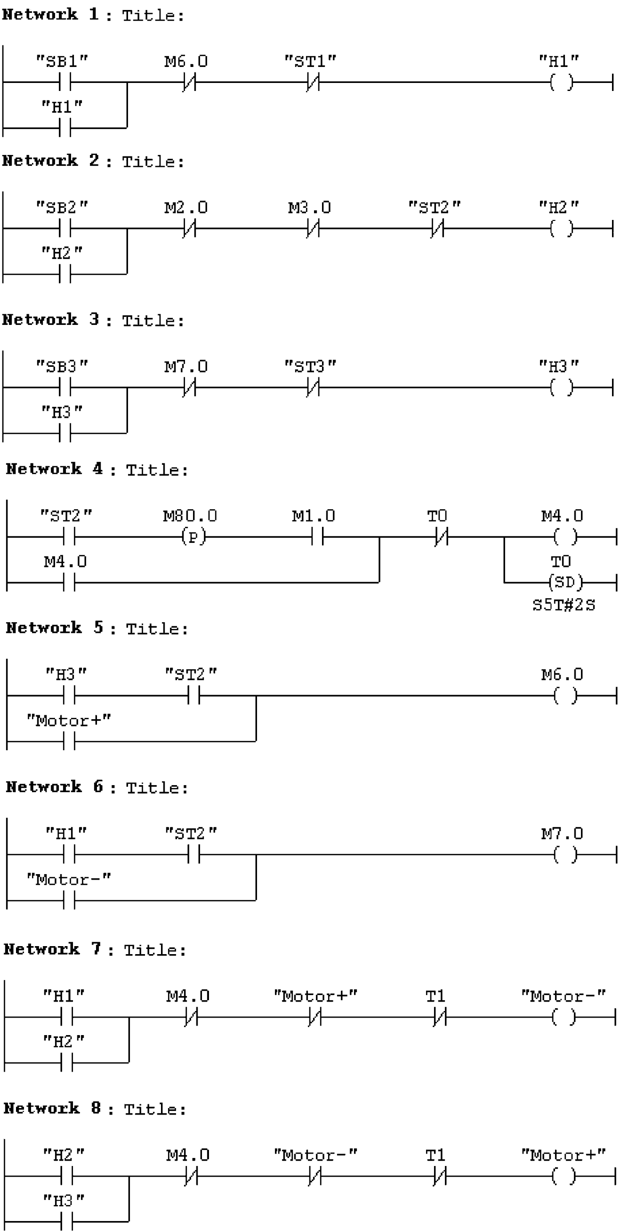
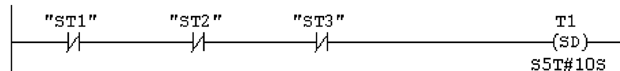
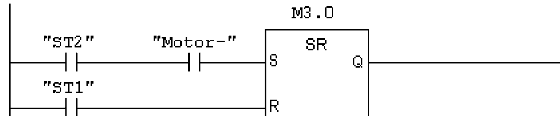


图 8-73　梯形图程序

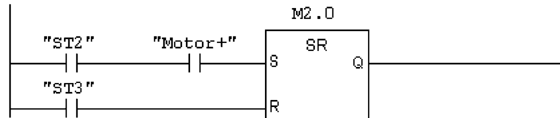
Network 9 : Title:



Network 10 : Title:



Network 11 : Title:



Network 12 : Title:

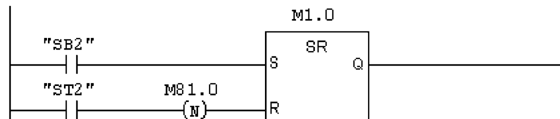


图 8-73　梯形图程序（续）

例二十八、霓虹灯广告屏控制器的设计

用 PLC 对霓虹灯广告屏（见图 8-74）实现控制，其具体要求如下：

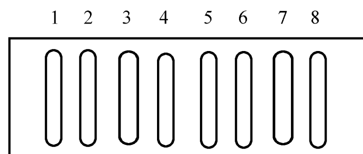


图 8-74　霓虹灯广告屏

该广告屏中间 8 个灯管亮灭的时序为第 1 根亮→第 2 根亮→第 3 根亮→…→第 8 根亮，时间间隔为 1s，全亮后，显示 10s，再反过来从 8→7→…→1 顺序熄灭。全灭后，停亮 2s，再从第 8 根灯管开始亮起，顺序点亮 7→6→…→1，时间间隔为 1s，显示 20s，再从1→2→…→8 顺序熄灭。全熄灭后，停亮 2s，再从头开始运行，周而复始。梯形图程序如图 8-75 所示。

Network 1: Title:

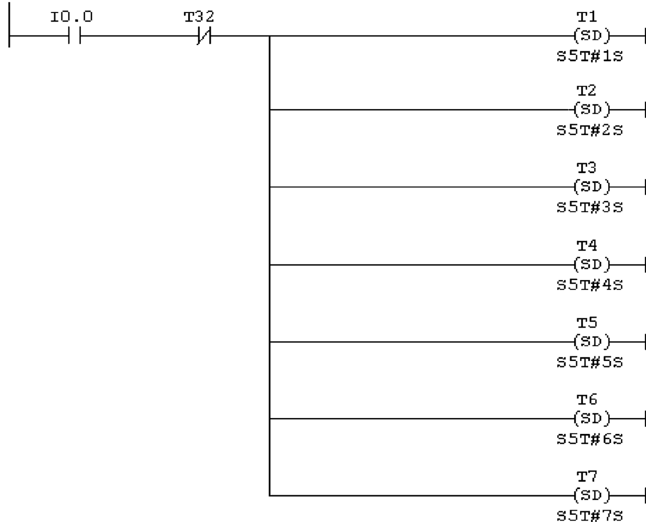
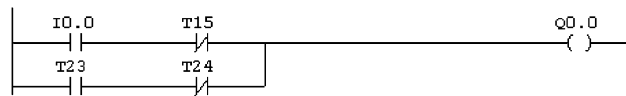
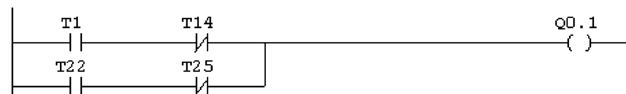


图 8-75　梯形图程序

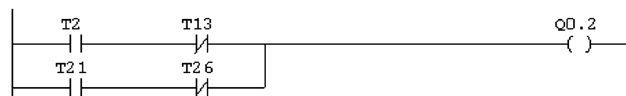
Network 2 : Title:



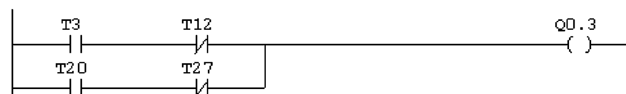
Network 3 : Title:



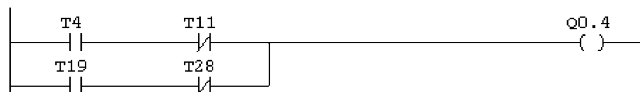
Network 4 : Title:



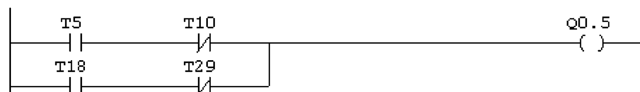
Network 5 : Title:



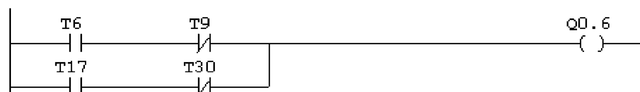
Network 6 : Title:



Network 7 : Title:



Network 8 : Title:



Network 9 : Title:

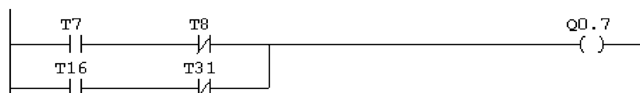
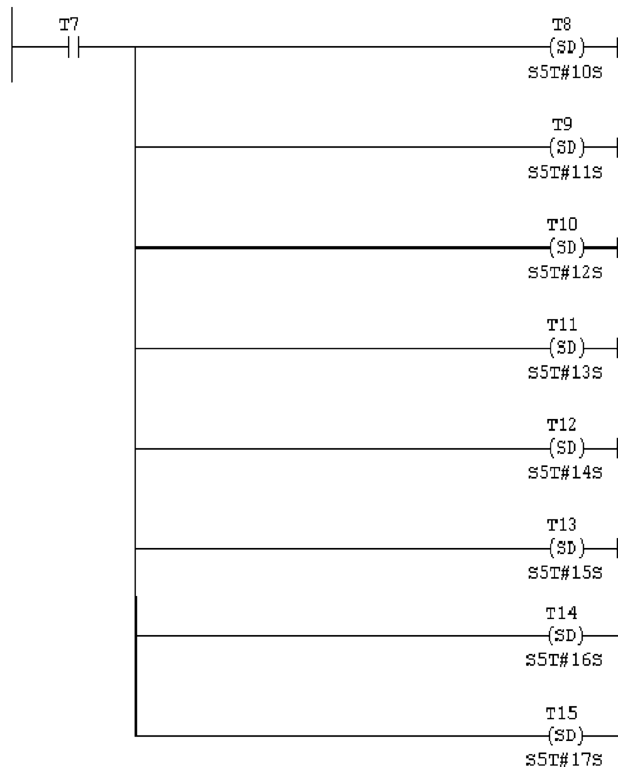


图 8-75　梯形图程序（续）


Network 10: Title:
T7
T8
SD
S5T#10S
T9
SD
S5T#11S
T10
SD
S5T#12S
T11
SD
S5T#13S
T12
SD
S5T#14S
T13
SD
S5T#15S
T14
SD
S5T#16S
T15
SD
S5T#17S

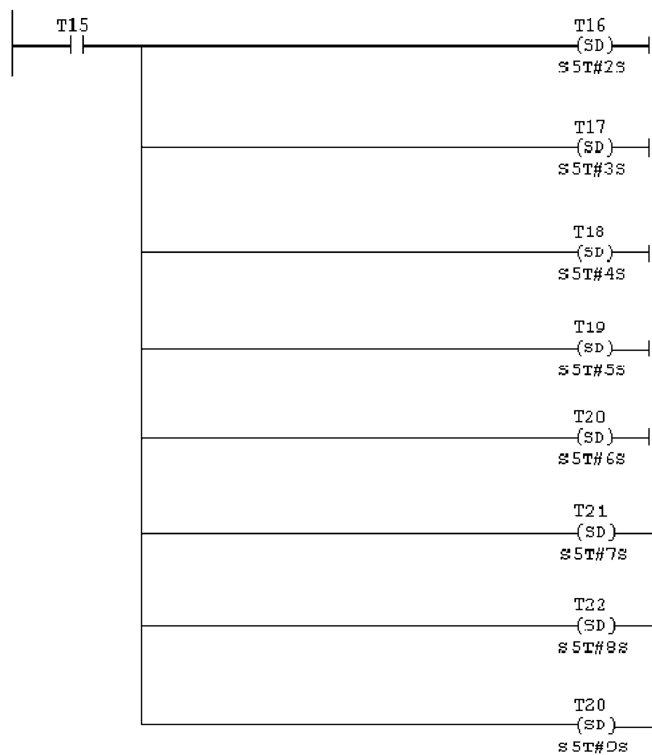

Network 11: Title:
T15
T16
SD
S5T#2S
T17
SD
S5T#3S
T18
SD
S5T#4S
T19
SD
S5T#5S
T20
SD
S5T#6S
T21
SD
S5T#7S
T22
SD
S5T#8S
T20
SD
S5T#9S

图 8-75　梯形图程序（续）

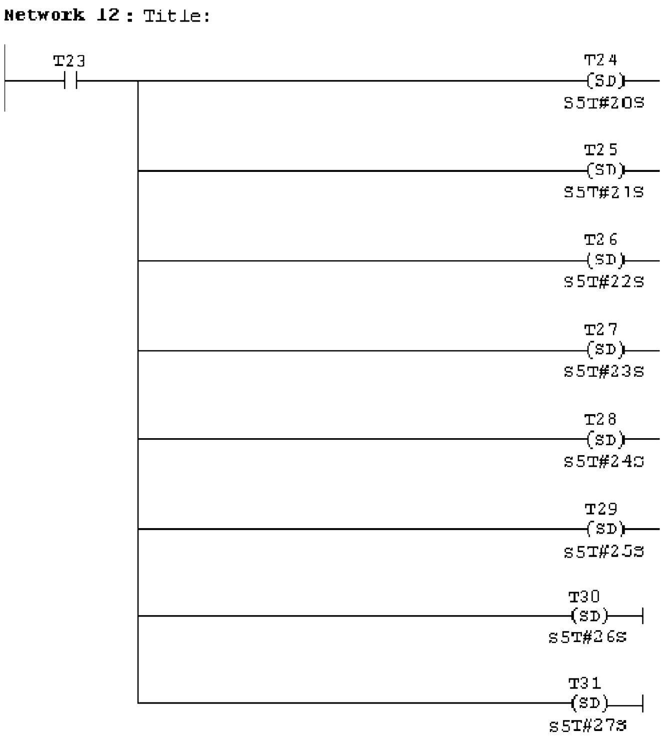


图 8-75　梯形图程序（续）

例二十九、自动售货机的 PLC 控制

如图 8-76 所示的自动售货机示意图，其工作要求如下。

1）此售货机可投入 1 元、5 元或 10 元硬币。

2）当投入的硬币总值超过 12 元时，汽水按钮指示灯亮；当投入的硬币总值超过 15 元时，汽水及咖啡按钮指示灯都亮。

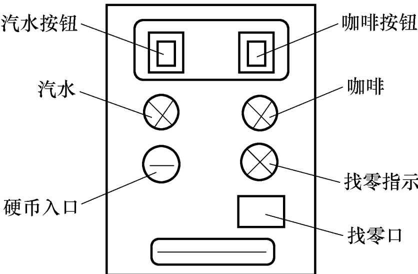


图 8-76　自动售货机示意图

3）当汽水按钮灯亮时，按汽水按钮，则汽水排出 7s 后自动停止，这段时间内，汽水指示灯闪动。

4）当咖啡按钮灯亮时，按咖啡按钮，则咖啡排出 7s 后自动停止，这段时间内，咖啡指示灯闪动。

5）若投入硬币总值超过按钮所需的钱数（汽水 12 元，咖啡 15 元）时，找钱指示灯亮，表示找钱动作，并退出多余的钱。

梯形图程序如图 8-77 所示。

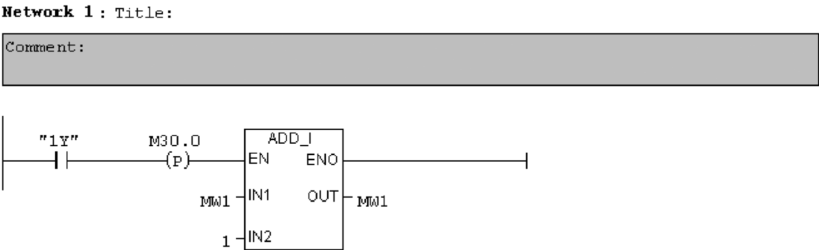
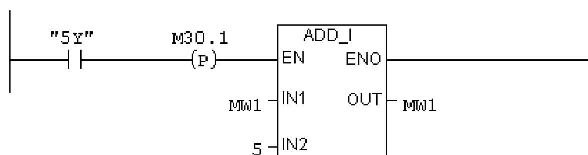


图 8-77　梯形图程序

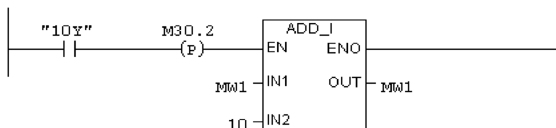
Network 2 : Title:

Comment:



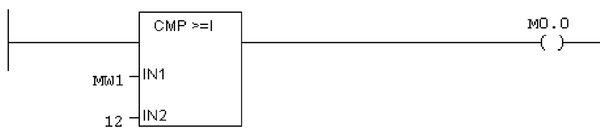
Network 3 : Title:

Comment:



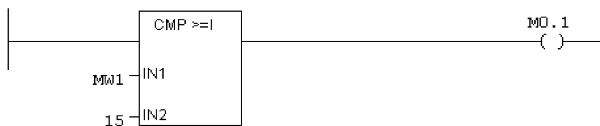
Network 4 : Title:

Comment:



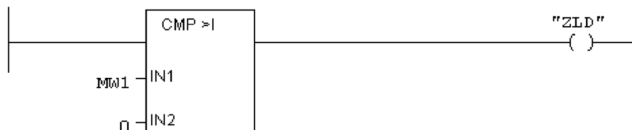
Network 5 : Title:

Comment:



Network 6 : Title:

Comment:



Network 7 : Title:

Comment:

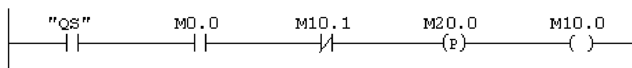
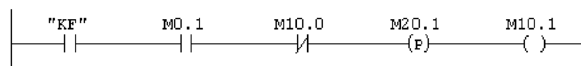


图 8-77　梯形图程序（续）

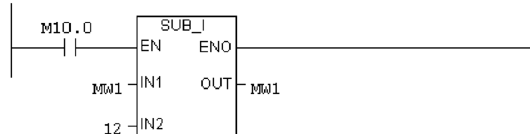
Network 8 : Title:

Comment:



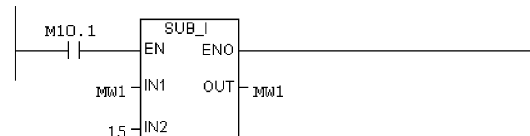
Network 9 : Title:

Comment:



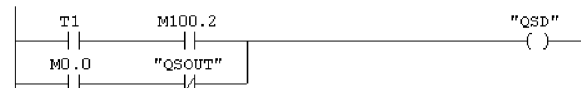
Network 10 : Title:

Comment:



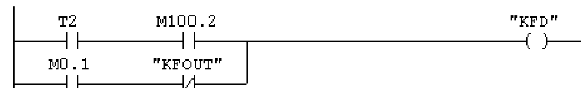
Network 11 : Title:

Comment:



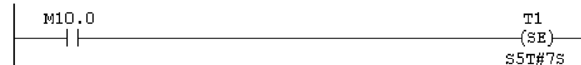
Network 12 : Title:

Comment:



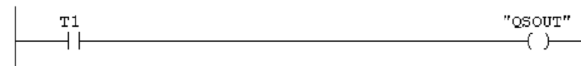
Network 13 : Title:

Comment:



Network 14 : Title:

Comment:



Network 15 : Title:

Comment:



图 8-77　梯形图程序（续）

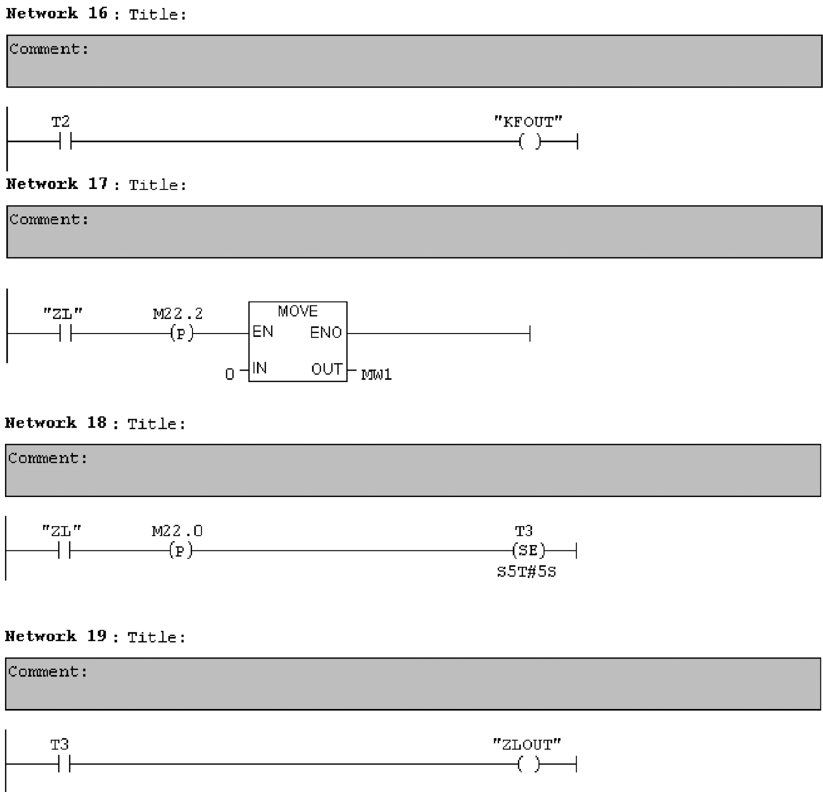


图 8-77　梯形图程序（续）

例三十、LED 数码显示 PLC 控制

利用 PLC 来控制一位七段 LED 数码管的显示，数码管的每一段都对应 PLC 的一个输出端子，PLC 输出端子的“1”、“0”状态对应于相应段的亮与灭。有两个按钮“＋”、“－”，每按动一次“＋”按钮，数字加 1，每按动一次“－”按钮，数字减 1。要求能正确显示数字 0～9 即可。七段转换表见表 8-13。

表 8-13　七段转换表

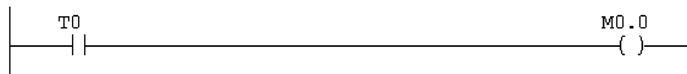
待变换的数据		七段显示	用于七段显示的 8bit 数据								七段显示
十六进制	二进制	的组成		g	f	e	d	c	b	a	
H0	0000		0	0	1	1	1	1	1	1	0
H1	0001		0	0	0	0	0	1	1	0	1
H2	0010		0	1	0	1	1	0	1	1	2
H3	0011		0	1	0	0	1	1	1	1	3
H4	0100		0	1	1	0	0	1	1	0	4
H5	0101		0	1	1	0	1	1	0	1	5
H6	0110		0	1	1	1	1	1	0	1	6
H7	0111		0	0	0	0	0	1	1	1	7
H8	1000		0	1	1	1	1	1	1	1	8
H9	1001		0	1	1	0	1	1	1	1	9
HA	1010		0	1	1	1	0	1	1	1	A
HB	1001		0	1	1	1	1	1	0	0	b
HC	1100		0	0	1	1	1	0	0	1	C
HD	1101		0	1	0	1	1	1	1	0	d
HE	1110		0	1	1	1	1	0	0	1	E
HF	1111		0	1	1	1	0	0	0	1	F

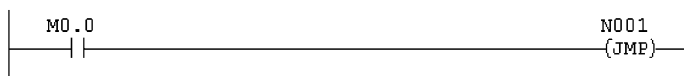
例三十一、时钟脉冲发生器

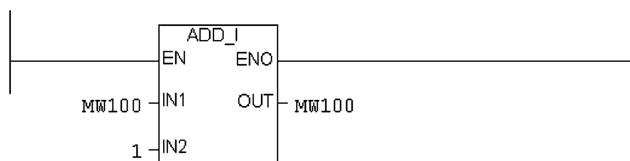
使用定时器实现自由设定时钟脉冲发生器功能（脉冲占空系数 1:1）。

梯形图程序如图 8-78 所示。









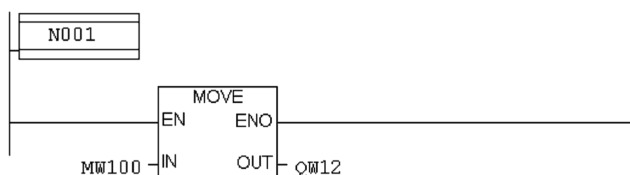


图 8-78　梯形图程序

存储字节 MB101 和 MB100 的单个位获得的频率见表 8-14。

表 8-14　存储字节 MB101 和 MB100 的单个位获得的频率

MB101/MB100	频率/Hz	持续时间/s	MB101/MB100	频率/Hz	持续时间/s
M101. 0	2. 0	0. 5	M100. 0	0. 0078125	128
M101. 1	1. 0	1	M100. 1	0. 0039062	256
M101. 2	0. 5	2	M100. 2	0. 0019531	512
M101. 3	0. 25	4	M100. 3	0. 0009765	1024
M101. 4	0. 125	8	M100. 4	0. 0004882	2048
M101. 5	0. 0625	16	M100. 5	0. 0002441	4096
M101. 6	0. 03125	32	M100. 6	0. 000122	8192
M101. 7	0. 015625	64	M100. 7	0. 000061	16384

存储字节 MB101 各位的信号状态见表 8-15。

表 8-15　存储字节 MB101 各位的信号状态

扫描周期	7	6	5	4	3	2	1	0	时间值/ms
0	0	0	0	0	0	0	0	0	250
1	0	0	0	0	0	0	0	1	250
2	0	0	0	0	0	0	1	0	250
3	0	0	0	0	0	0	1	1	250
4	0	0	0	0	0	1	0	0	250
5	0	0	0	0	0	1	0	1	250
6	0	0	0	0	0	1	1	0	250
7	0	0	0	0	0	1	1	1	250
8	0	0	0	0	1	0	0	0	250
9	0	0	0	0	1	0	0	1	250
10	0	0	0	0	1	0	1	0	250
11	0	0	0	0	1	0	1	1	250
12	0	0	0	0	1	1	0	0	250

M101.1 的信号状态如图 8-79 所示。

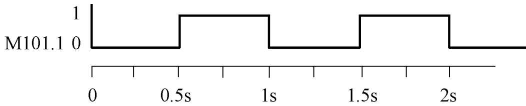


图 8-79　M101.1 的信号状态

例三十二、4 个灯按不同周期闪烁

当按起动按钮时，L1 以 0.5s 的周期闪烁，L2 以 1s 的周期闪烁，L3 以 2s 的周期闪烁，L4 以 4s 的周期闪烁，按下停止按钮，所有灯熄灭。

梯形图程序如图 8-80 所示。

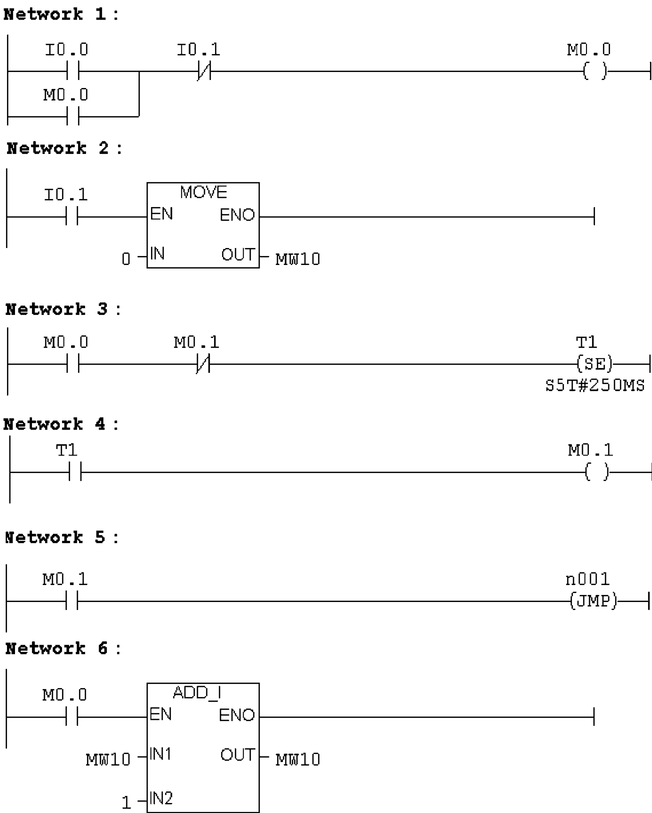


图 8-80　梯形图程序

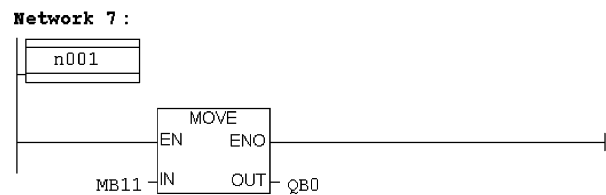


图 8-80 梯形图程序（续）

例三十三、加热炉

操作员按起动按钮开始加热如图 8-81 所示的加热炉。操作员能够使用如图 8-81 所示的拨码开关设定加热时间。操作员设定的值以 BCD 格式用秒单位显示。

加热系统的元件和相应的绝对地址见表 8-16。

梯形图程序如图 8-82 所示。

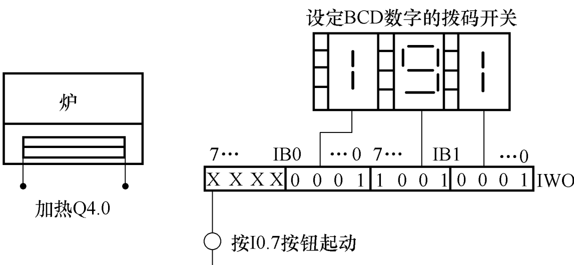


图 8-81 加热炉

表 8-16 加热系统的元件和相应的绝对地址

系统元件	绝对地址	系统元件	绝对地址
起动按钮	I0. 7	百位数拨码开关	I0. 0 ~ I0. 3
个位数拨码开关	I1. 0 ~ I1. 3		
十位数拨码开关	I1. 4 ~ I1. 7	开始加热	Q4. 0

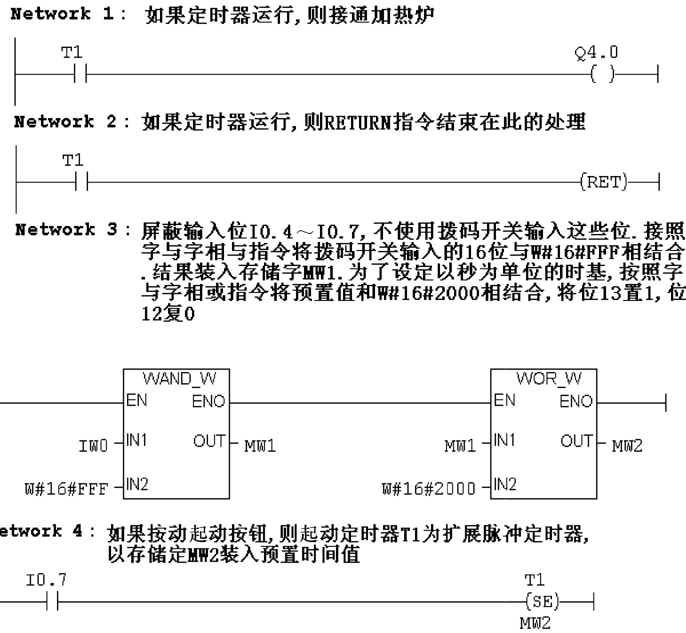


图 8-82 梯形图程序

例三十四、彩灯控制

当按下起动按钮时，彩灯 L1、L2 同时亮；过 1s 后，L1 熄灭，L2 保持亮；过 1s 后，L1、L2 同时灭；过 1s 后，L1 亮，L2 保持灭；再过 1s 后，L1、L2 又同时亮，如此循环闪烁，直到按下

停止按钮，彩灯工作终止。梯形图程序如图 8-83 所示。

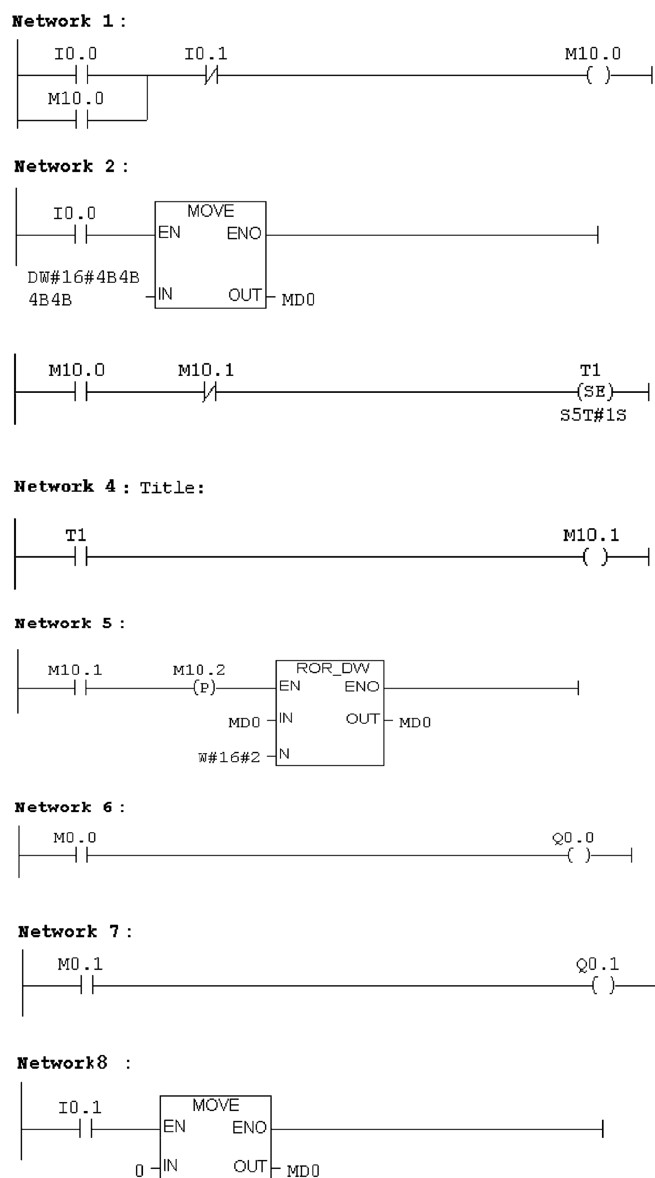


图 8-83 梯形图程序

例三十五、双缸顺序动作回路 A1B1B0A0

双缸顺序动作示意图如图 8-84 所示。

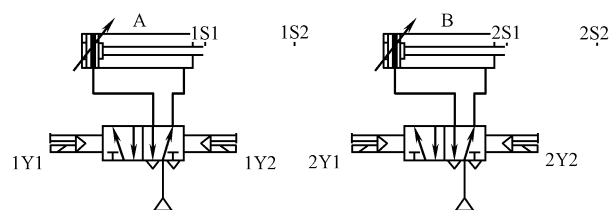


图 8-84 双缸顺序动作示意图

梯形图程序如图 8-85 所示。

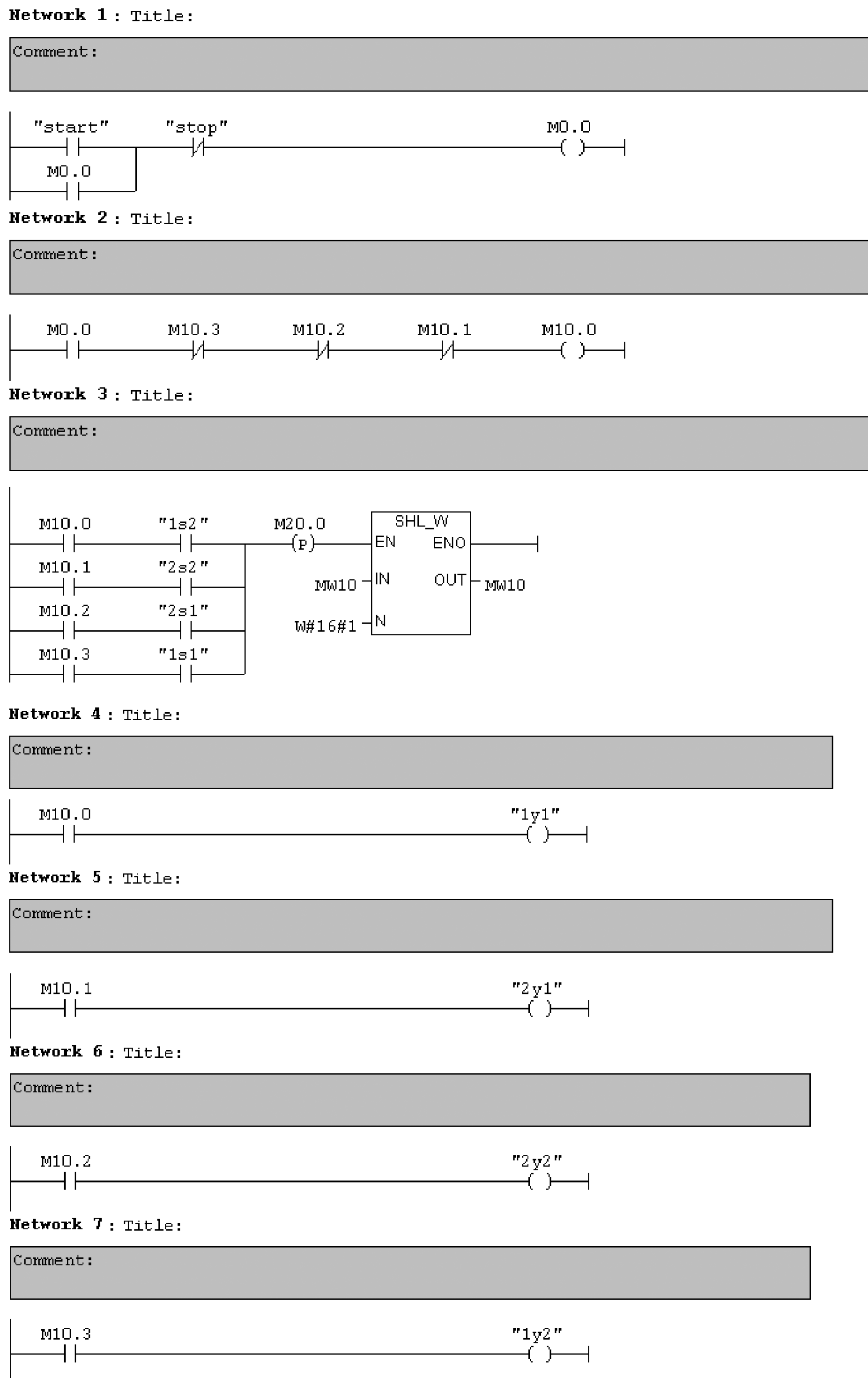


图 8-85 梯形图程序

例三十六、一个按钮控制 4 个灯

按下开关 I0.0，L1、L2、L3、L4 依次亮灭，周而复始，时间间隔为 1s（即 L1 亮 1s 后灭，

接着 L2 亮，如此循环)。梯形图程序如图 8-86 所示。

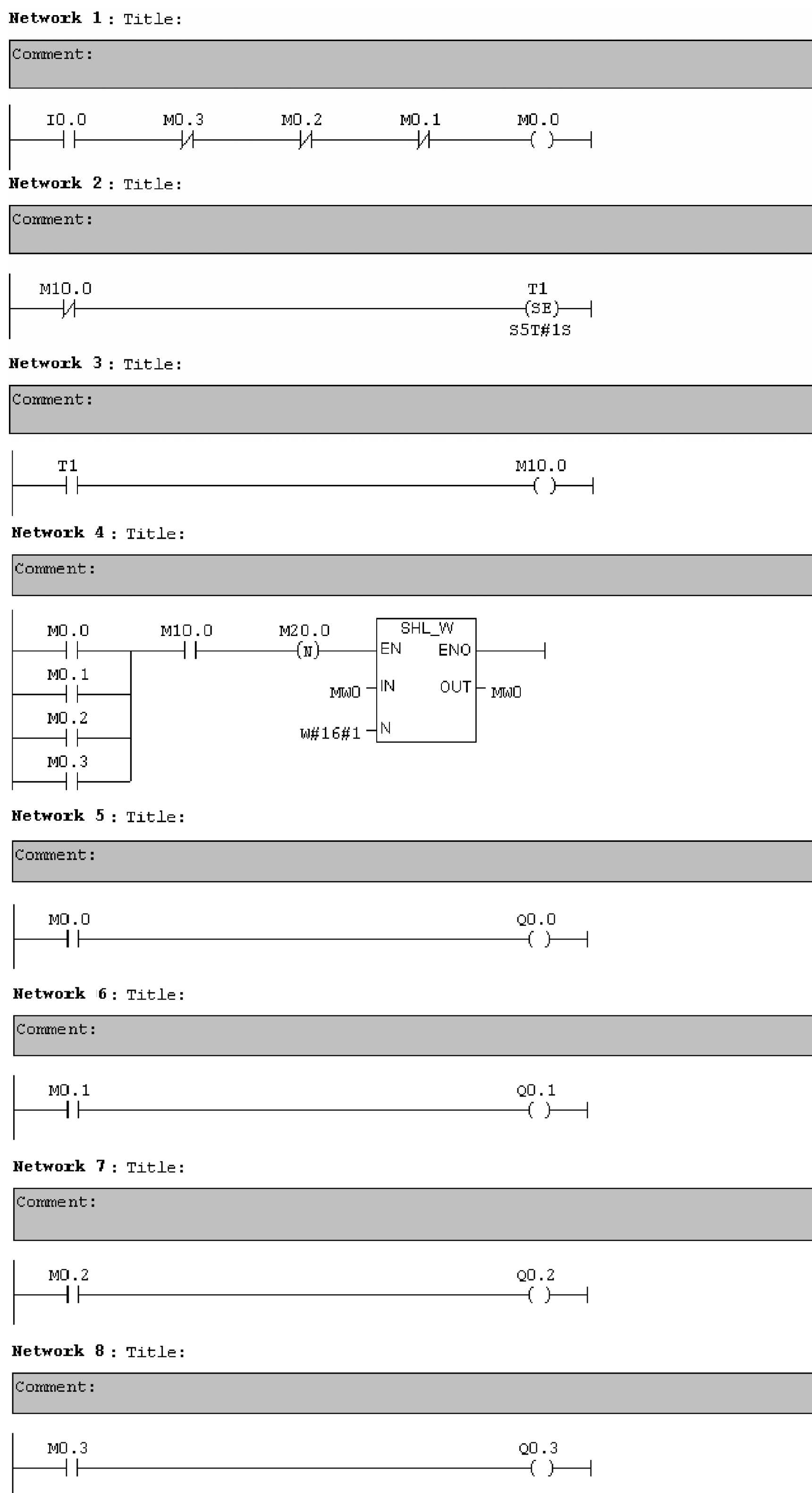


图 8-86 梯形图程序

例三十七、控制 3 个霓虹灯闪烁

试编写用 PLC 控制 3 个霓虹灯闪烁的程序。工作要求如下：

1）首先 A 灯亮。

2）1s 后 A 灯灭，B 灯亮。

3）再过 1s 后 B 灯灭，C 灯亮。

4）再过 1s 后 C 灯灭。

5）再过 1s 后，A、B、C 三灯全亮。

6）再过 1s 后，A、B、C 三灯全灭。

7）再过 1s 后，A、B、C 三灯全亮。

8）再过 1s 后，A、B、C 三灯全灭。

然后 1）~8）步重复循环。要求用一个开关控制，当它闭合接通时霓虹灯工作，断开时停止工作。

梯形图程序如图 8-87 所示。

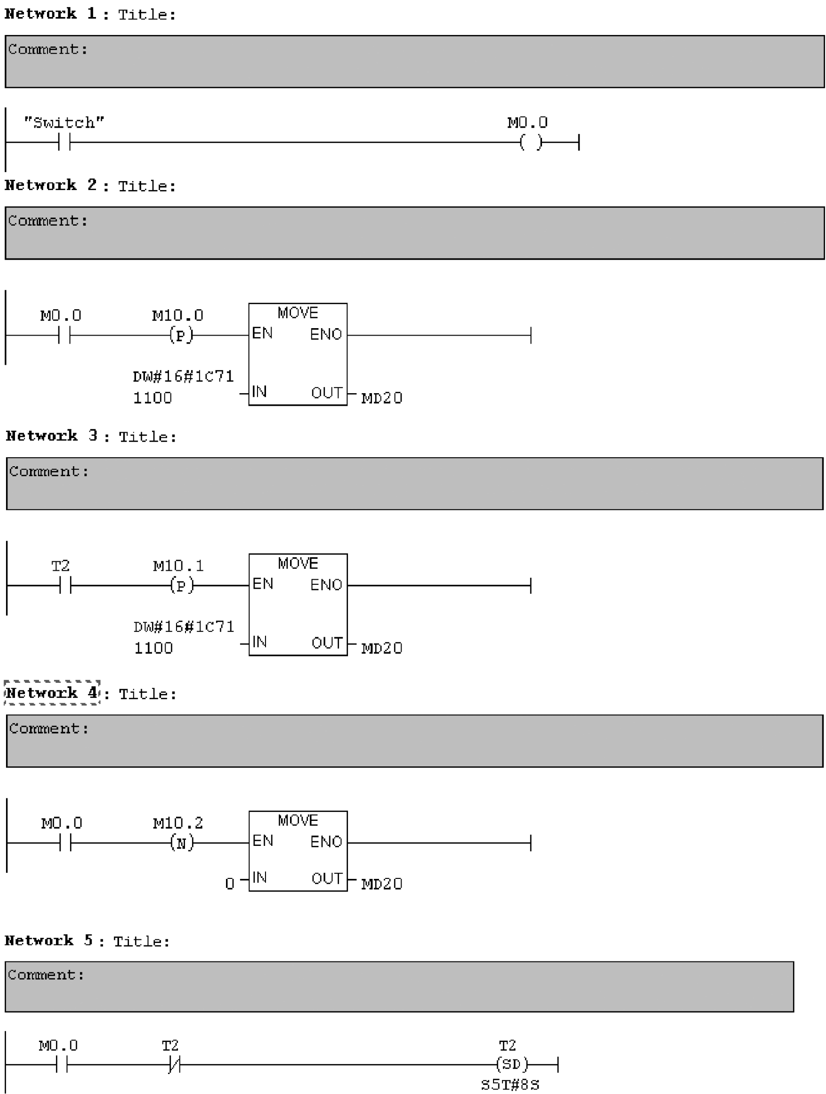
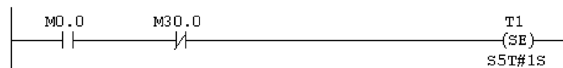


图 8-87　梯形图程序

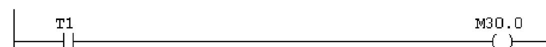
Network 6 : Title:

Comment:



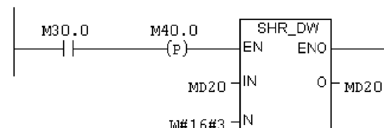
Network 7 : Title:

Comment:



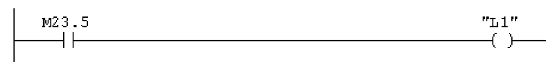
Network 8 : Title:

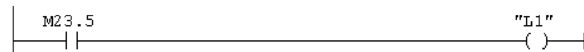
Comment:



Network 9 : Title:

Comment:





Network 10 : Title:

Comment:



Network 11 : Title:

Comment:

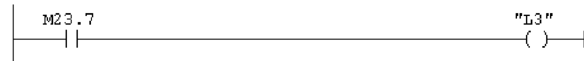


图 8-87　梯形图程序（续）

二、运料小车控制系统

运料小车是工厂原料生产车间的主要设备之一，如何实现运料小车的自动控制，使整个生产流程实现自动化，从而加快生产速度，提高生产效率，是现在工厂需要解决的问题之一。

1. 系统工作原理

本系统使用 PLC 300 系统来实现运料小车的自动控制，其系统原理图如图 8-88 所示。

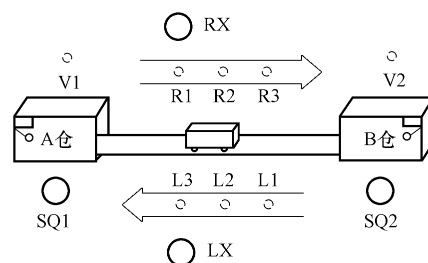


图 8-88　系统原理图

当合上起动按钮 SD（参照图 8-89）后，行程开关 SQ1 处于接通状态，小车停在原位（A 仓），装料指示灯 V1 点亮，开始向小车装料。延时 5s 后 V1 熄灭，装料完毕。打开右行开关 RX，小车开始右行，R1、R2、R3 顺序点亮。到达 B 仓后，行程开关 SQ2 接通，卸料指示灯 V2 点亮，小车开始卸料。延时 5s 后 V2 熄灭，卸料完毕。打开左行开关 LX，小车开始左行，L1、L2、L3 顺序点亮。到达 A 仓后，行程开关 SQ1 接通，小车回到原位，流程一直循环，从而实现对运料小车的控制。

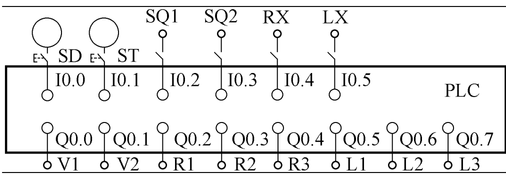


图 8-89　PLC 接线图

2. PLC 接线图及 I/O 点分配

（1）PLC 的接线图（见图 8-89）

（2）I/O 分配表

PLC 的 I/O 分配表见表 8-17。

表 8-17　I/O 点代码和地址编号

名　称	代码	地址编号	名　称	代码	地址编号
输入信号			输出信号		
起动输入	SD	I0. 0	装料输出	1 V1	Q4. 0
停止输入	ST	I0. 1	卸料输出	2 V2	Q4. 1
限制向右运动输入	SQ1	I0. 2	右行输出过程第一点	3 R1	Q4. 2
限制向左运动输入	SQ2	I0. 3	右行输出过程第二点	4 R2	Q4. 3
向右运动输入	RX	I0. 4	右行输出过程第三点	5 R3	Q4. 4
向左运动输入	LX	I0. 5	左行输出过程第一点	6 L1	Q4. 5
			左行输出过程第二点	7 L2	Q4. 6
			左行输出过程第三点	8	Q4. 7

3. 运料小车控制系统源程序

运料小车控制系统源程序如图 8-90 所示。

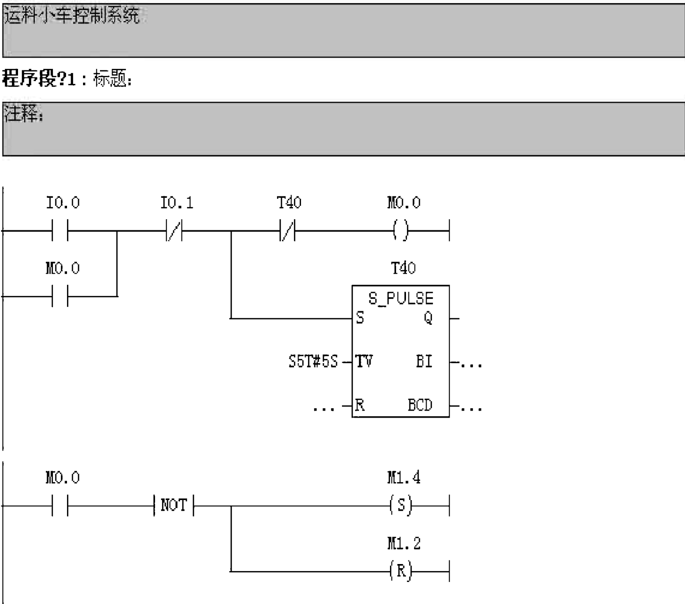


图 8-90　运料小车控制系统源程序

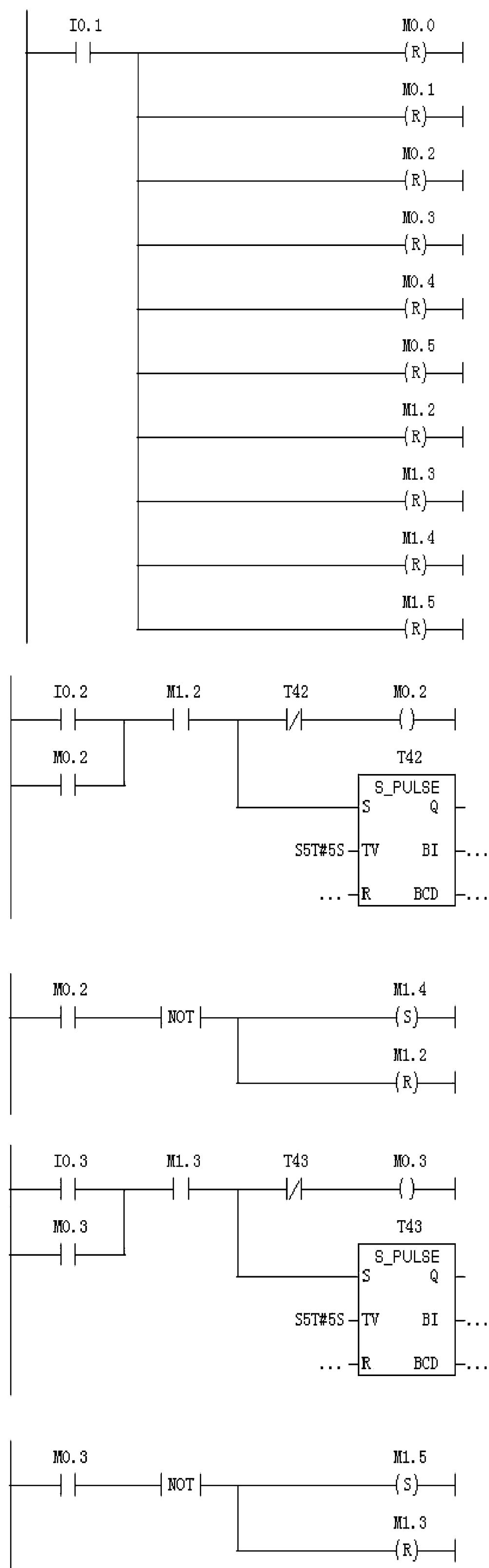


图 8-90 运料小车控制系统源程序（续）

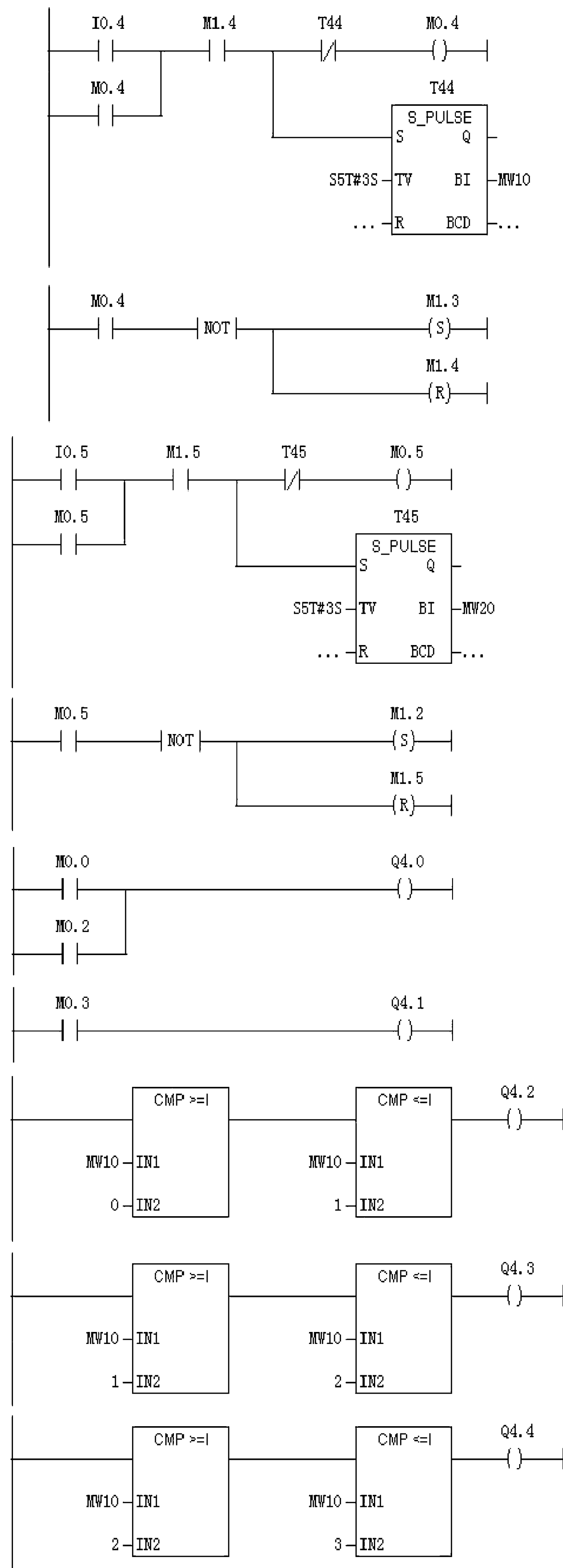


图 8-90　运料小车控制系统源程序（续）

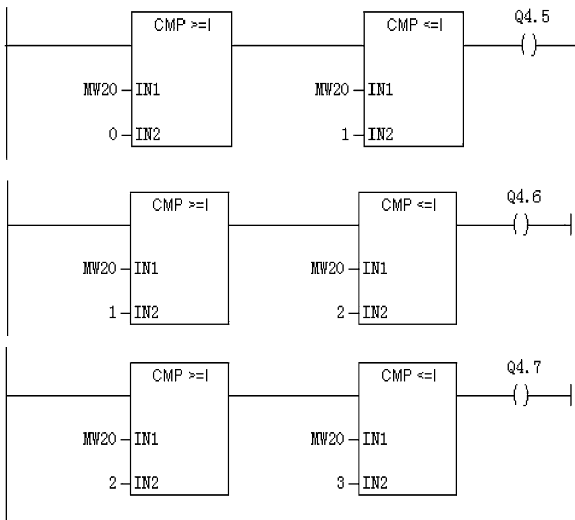


图 8-90 运料小车控制系统源程序（续）

三、水塔水位控制

在自动供水系统或大型泵站供水系统中，可以利用 PLC 实现供水的自动化和泵站的水位控制，从而达到提高供水效率，节约用水，节约劳动力和成本的目的。

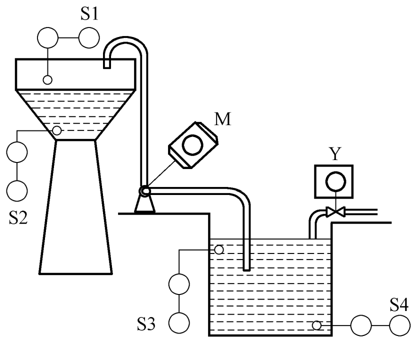


图 8-91 系统工作原理图

1. 系统工作原理

系统工作原理图如图 8-91 所示。

当供水池水位低于水位界限时（用图中的 S4），阀门 Y 打开给水池注水（Y 为 ON），同时定时器开始计时；2s 后，如果 S4 继续保持 0FF 状态，那么阀门 Y 的指示灯开始以 0.5s 的时间间隔闪烁，表示阀门 Y 没有进水，出现故障；当水池水位到达高水位界限时 S3 打开（ON），阀门 Y 关闭（OFF）。

当 S4 为 ON 时，如果水塔水位低于低水位界限（S2 为 0FF），水泵 M 开始从供水池中抽水，当水塔水位达到高水位界限时（S1 为 ON），水泵 M 停止供水。

2. PLC 接线图及 I/O 点分配

（1）PLC 的接线图（见图 8-92）

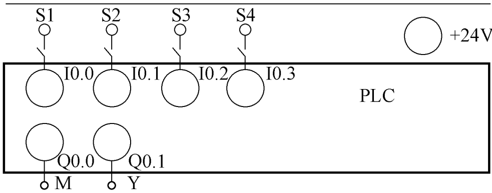


图 8-92 PLC 的接线图

（2）I/O 分配表

PLC 的 I/O 分配表见表 8-18。

表 8-18 I/O 点代码和地址编号

名　称	代码	地址编号	名　称	代码	地址编号
输入信号			液面传感器	S4	I0.3
液面传感器	S1	I0.0	输出信号		
液面传感器	S2	I0.1	水泵状态输出	1　M	Q4.0
液面传感器	S3	I0.2	供水阀状态输出	2　Y	Q4.1

3. 水塔水位控制系统源程序

水塔水位控制系统源程序如图 8-93 所示。

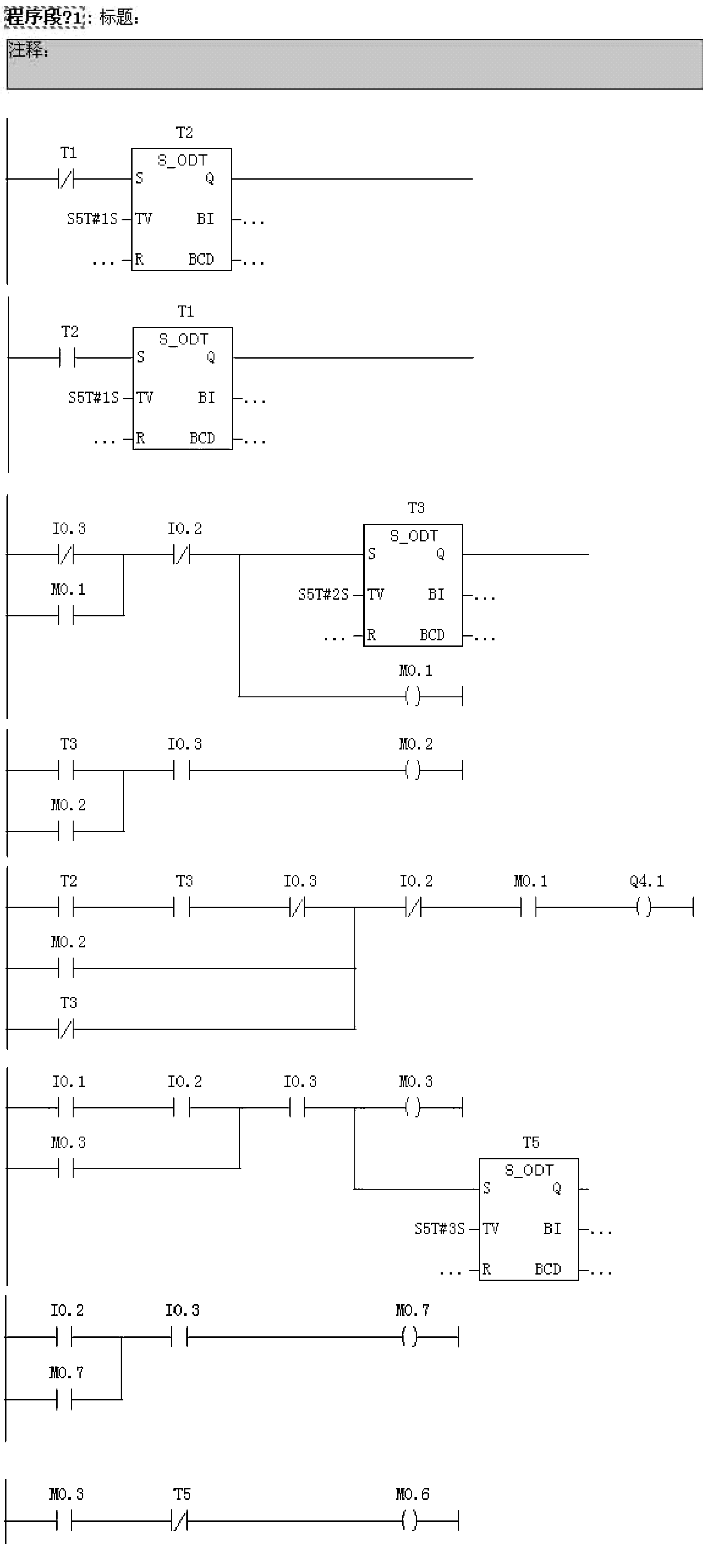


图 8-93　水塔水位控制系统源程序

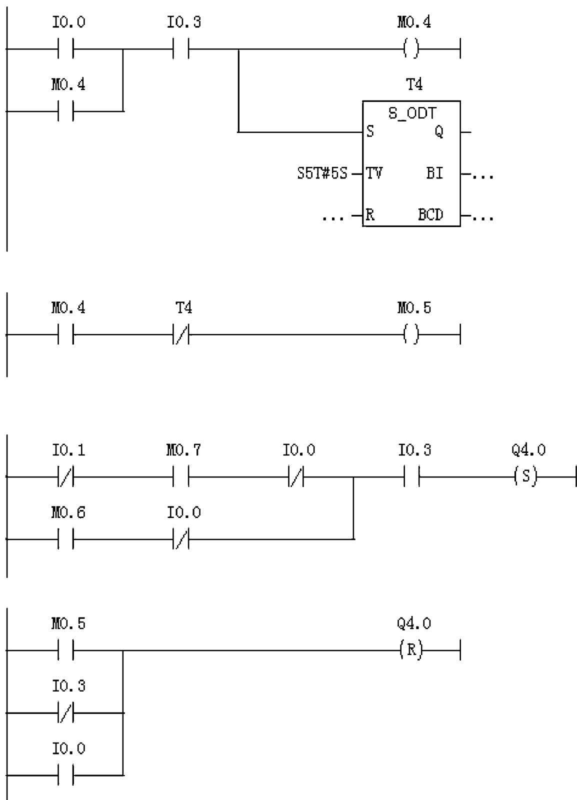


图 8-93 水塔水位控制系统源程序（续）

四、四节传送带控制系统

在工厂的生产线中，自动传输带是一种非常常见的设备，利用 PLC 来对自动传输带进行控制，在很早之前便实现了。本小节利用 S7 300 PLC 来实现对四节传送带的控制。

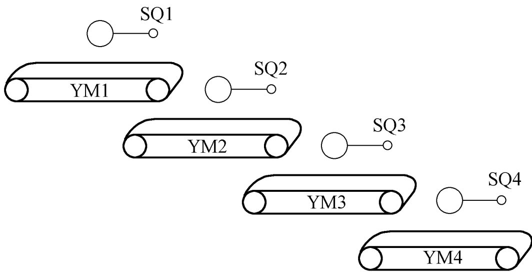


图 8-94 系统的工作原理图

1. 系统工作原理

系统的工作原理图如图 8-94 所示。

在四节传送带控制系统中，四条传送带分别用四台电动机带动，具体控制如下：起动时先起动最末一条传送带，再按逆流向依次起动其他传送带，在起动下一条传送带之前均应根据工艺要求设定延时时间，本程序设定为 5s。停止时应先停止最前一条传送带，待料运送完毕后依次停止其他传送带。当某条传送带发生故障时，该传送带机及其前面的传送带机立即停止，而该传送带后面的传送带运送完上面的物料后再停止运行。当某条传送带机上有重物时，该传送带机前面的传送带机停止转动，5s 后该传送带机停止转动，该传送带机后面的传送带机等到上面的物料运送完后才停止转动。

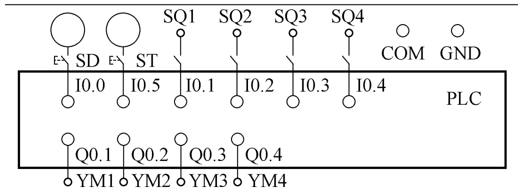


图 8-95 PLC 接线图

2. PLC 接线图及 I/O 点分配

（1）PLC 的接线图（见图 8-95）

（2）I/O 分配表

PLC 的 I/O 分配表见表 8-19。

表 8-19　I/O 点代码和地址编号

名　称	代码	地址编号	名　称	代码		地址编号
输入信号			负载或故障输入 4	SQ4		I0. 5
起动输入	SD	I0. 0	输出信号			
停止输入	ST	I0. 1	传送带机 1	1	KM1	Q4. 1
负载或故障输入 1	SQ1	I0. 2	传送带机 2	2	KM2	Q4. 2
负载或故障输入 2	SQ2	I0. 3	传送带机 3	3	KM3	Q4. 3
负载或故障输入 3	SQ3	I0. 4	传送带机 4	4	KM4	Q4. 4

3. 四节传送带控制系统源程序

四节传送带控制系统源程序如图 8-96 所示。

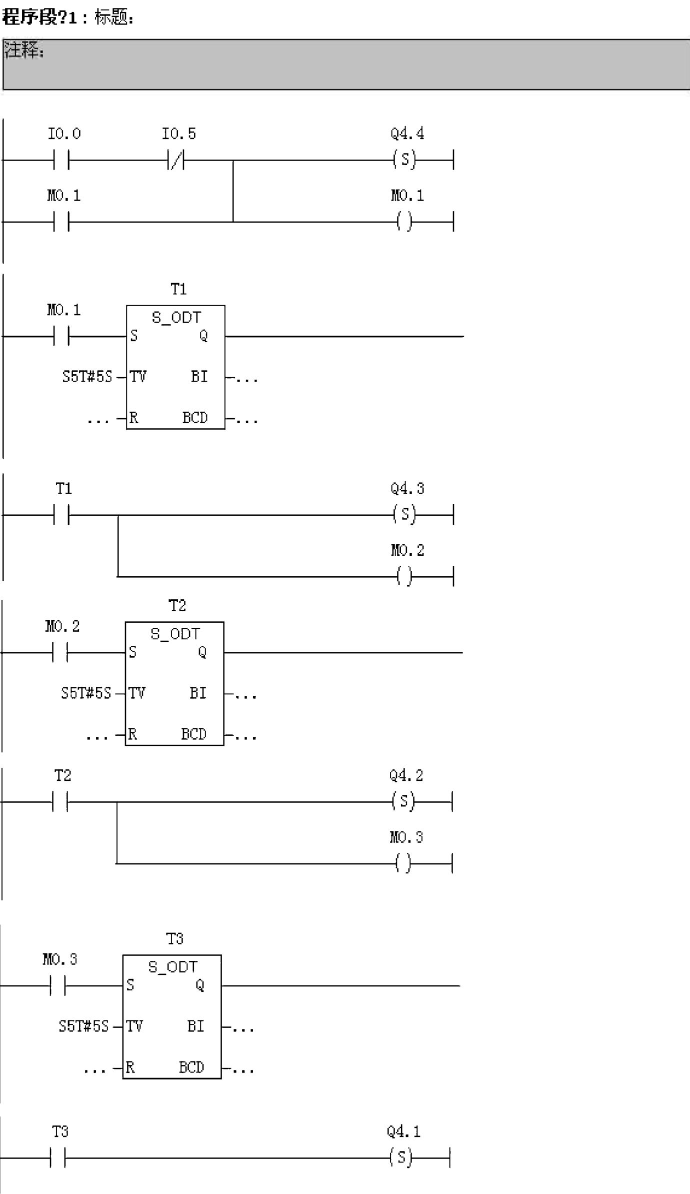


图 8-96　四节传送带控制系统源程序

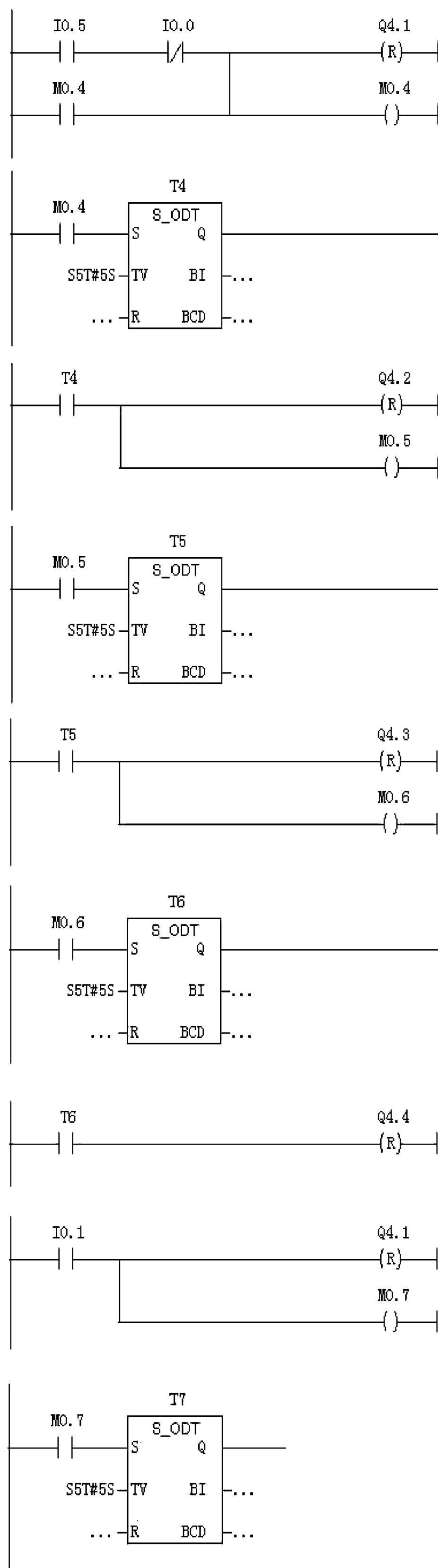


图 8-96 四节传送带控制系统源程序（续）

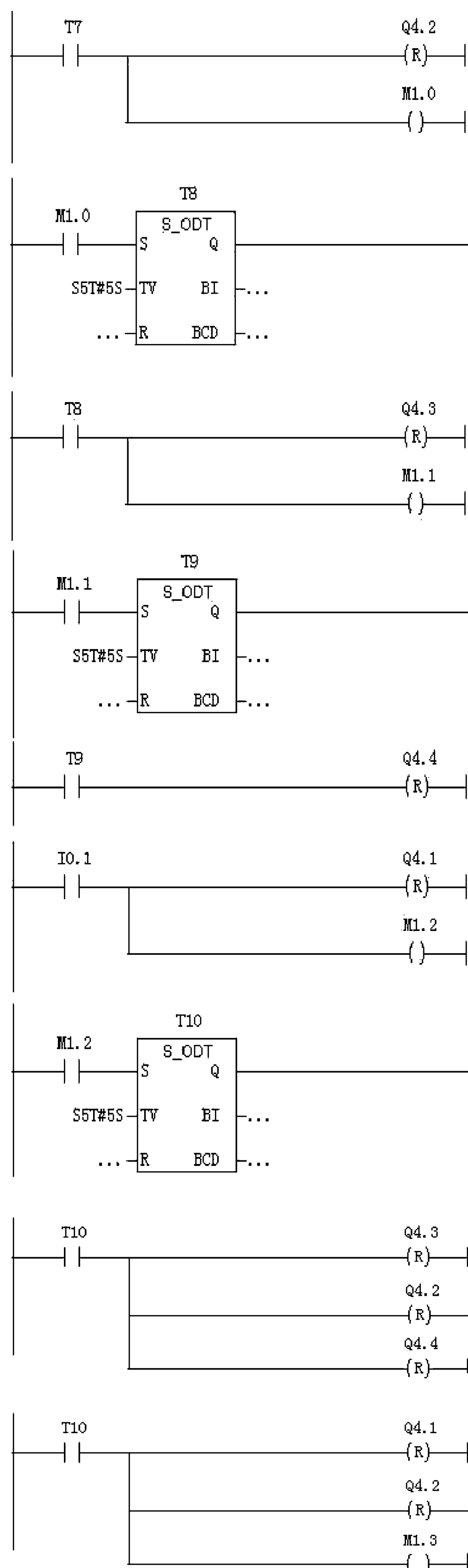


图 8-96　四节传送带控制系统源程序（续）

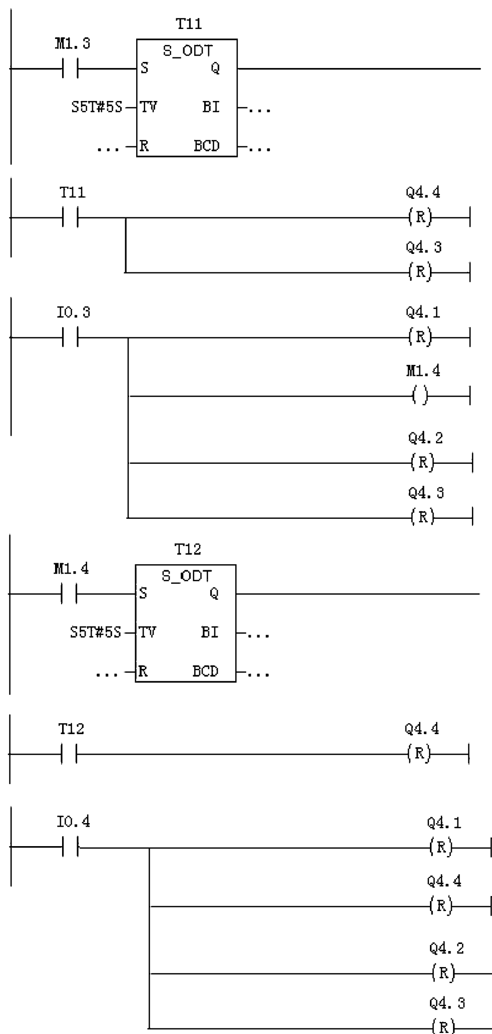


图 8-96　四节传送带控制系统源程序（续）

五、电梯控制系统

电梯的初始形态是一种升降设备，起源于古代农业和建筑业中的原始起重升降机械，现代电梯兴盛的根本原因是电力、电动机的作用。1887 年德国出现了电力拖动的升降机——电梯，到 1889 年纽约使用了由美国奥斯公司推出的真正的电梯。

PLC 最早是根据顺序逻辑控制的需要而发展起来的，是专门为工业环境应用而设计的数字运算操作的电子装置。鉴于其各种优点，目前，电梯的继电器控制方式已逐渐被 PLC 控制所代替。同时，由于电动机交流变频调速技术的发展，电梯的拖动方式已由原来的直流调速逐渐过渡到了交流变频调速。因此，PLC 控制技术加变频调速技术已成为现代电梯行业的一个热点。

1. 系统工作原理

图 8-97 所示是电梯 PLC 控制系统的基本结构图。电梯的安全保护装置，用于电梯的起停控制；轿厢操作盘用于控制轿厢门的关闭，轿厢需要达到的楼层等的控制；厅外呼叫的主要作用是：当有呼叫人员进行呼叫时，电梯能够准确达到呼叫位置；指层器用于显示电梯达到的具体位置；拖动控制用于控制电梯的起停、加速、减速等；门机控制主要用于控制当电梯达到一定位置

后，电梯门应该能够自动打开，或者门外有乘梯人员要求乘梯时，电梯门应该能够自动打开。电梯实物图如图 8-98 所示。

电梯信号控制基本由 PLC 软件实现。电梯 PLC 信号控制系统框图如图 8-99 所示，输入到 PLC 的控制信号有运行方式选择（如自动、有司机、检修、消防运行方式等）、运行控制、轿内指令、层站召唤、安全保护信息、旋转编码器光电脉冲、开关门及限位信号、门区和平层信号等。

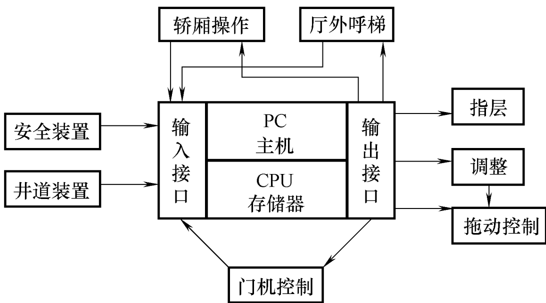


图 8-97　电梯 PLC 控制系统的基本结构图

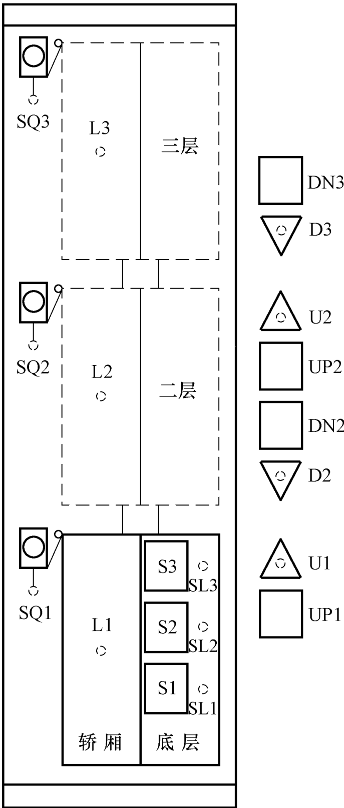


图 8-98　电梯实物图

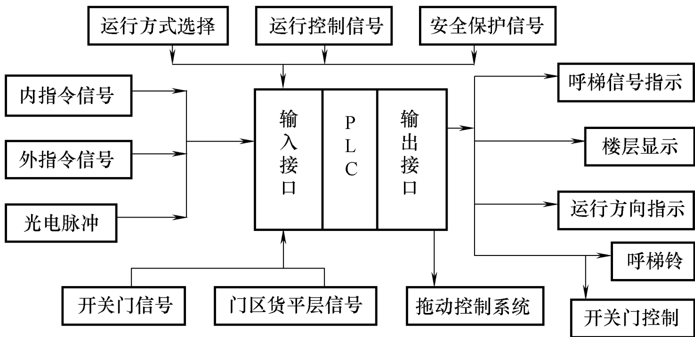


图 8-99　电梯 PLC 信号控制系统框图

2. PLC 接线图及 I/O 点分配

1）PLC 的接线图如图 8-100 所示。

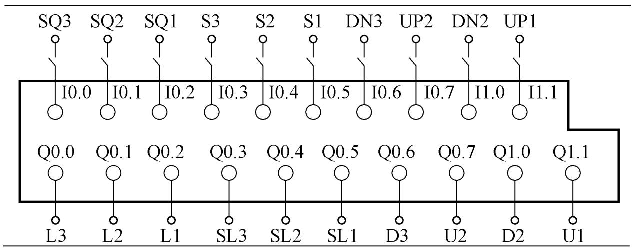


图 8-100　PLC 接线图

2）PLC 的 I/O 分配表见表 8-20。

表 8-20　I/O 点代码和地址编号

名称	代码	地址编号	名称	代码	地址编号
输入信号			输出信号		
三层行程开关	SQ3	I0. 0	三层指示灯	1　L3	Q4. 0
二层行程开关	SQ2	I0. 1	二层指示灯	2　L2	Q4. 1
一层行程开关	SQ1	I0. 2	一层指示灯	3　L1	Q4. 2
三层内选按钮	S3	I0. 3	三层内选指示	4　SL3	Q4. 3
二层内选按钮	S2	I0. 4	二层内选指示	5　SL2	Q4. 4
一层内选按钮	S1	I0. 5	一层内选指示	6　SL1	Q4. 5
三层下呼按钮	DN3	I0. 6	三层下呼指示	7　D3	Q4. 6
二层上呼按钮	UP2	I0. 7	二层上呼指示	8　U2	Q4. 7
二层下呼按钮	DN2	I1. 0	二层下呼指示	9　D2	Q5. 0
一层上呼按钮	UP1	I1. 1	一层上呼指示	10　U1	Q5. 1

3. 电梯控制系统源程序

电梯控制系统源程序如图 8-101 所示。

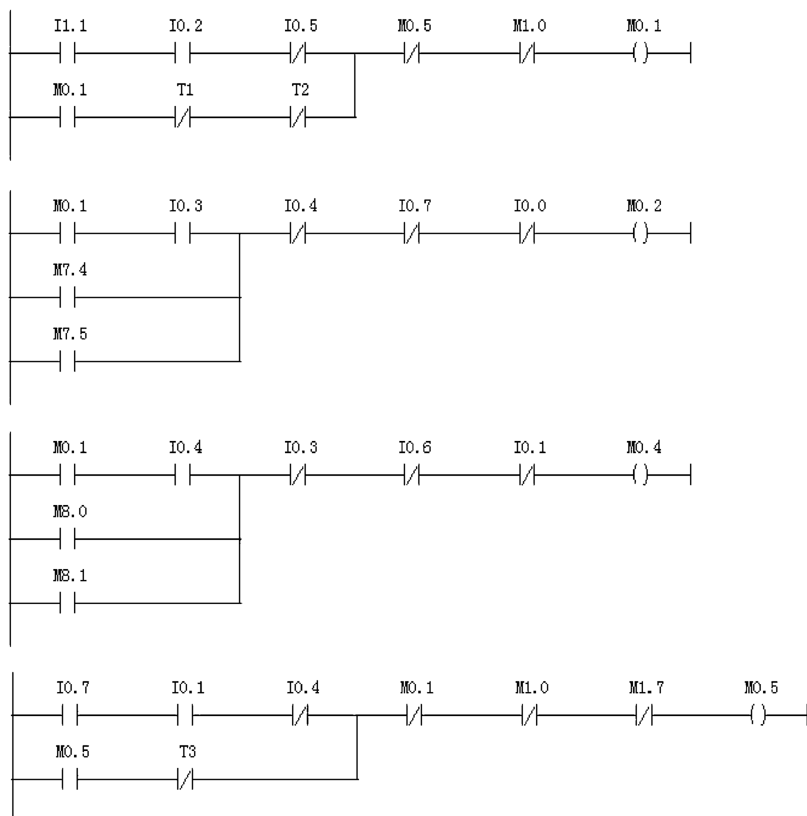


图 8-101　电梯控制系统源程序

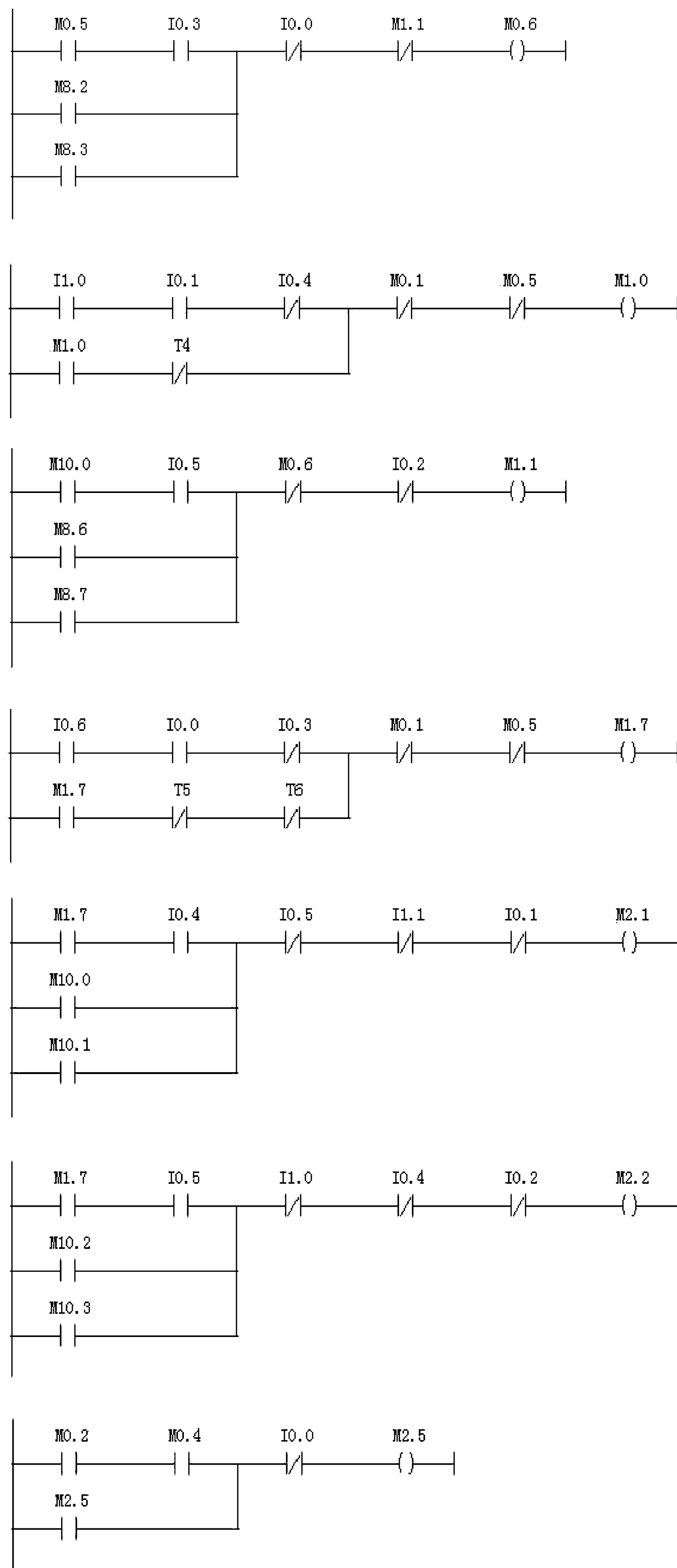


图 8-101　电梯控制系统源程序（续）

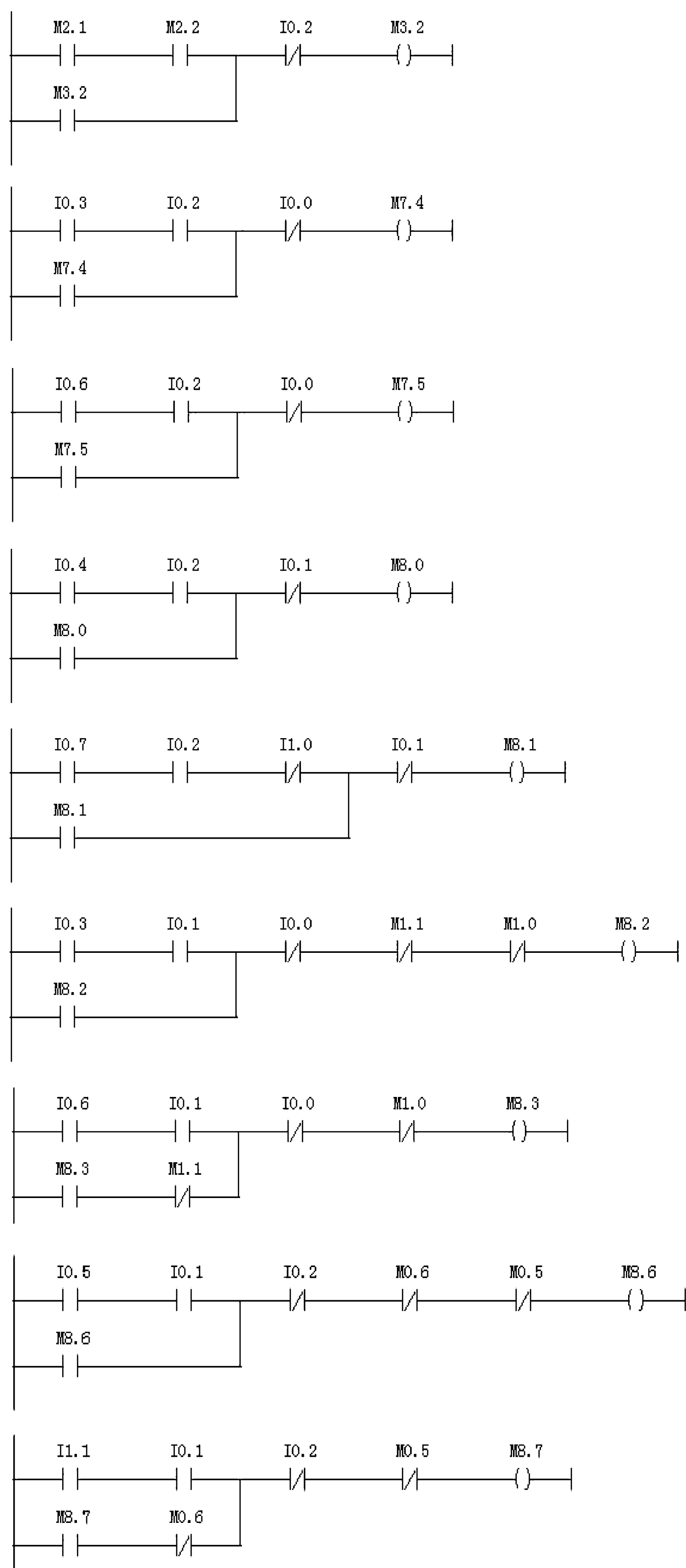
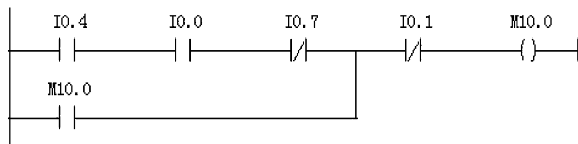
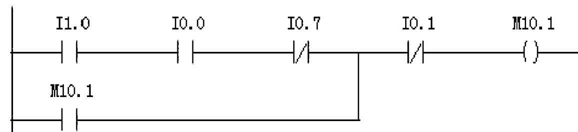
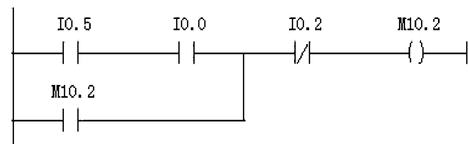
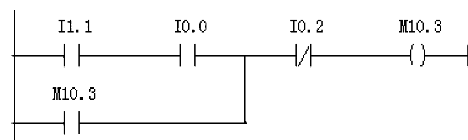

M2.1
M2.2
I0.2
M3.2
M3.2
I0.3
I0.2
I0.0
M7.4
M7.4
I0.6
I0.2
I0.0
M7.5
M7.5
I0.4
I0.2
I0.1
M8.0
M8.0
I0.7
I0.2
I1.0
I0.1
M8.1
M8.1
I0.3
I0.1
I0.0
M1.1
M1.0
M8.2
M8.2
I0.6
I0.1
I0.0
M1.0
M8.3
M8.3
M1.1
I0.5
I0.1
I0.2
M0.6
M0.5
M8.6
M8.6
I1.1
I0.1
I0.2
M0.5
M8.7
M8.7
M0.6

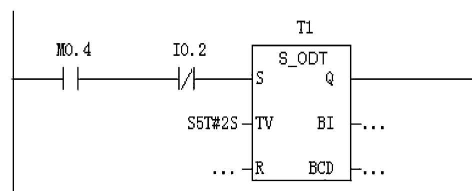
图 8-101 电梯控制

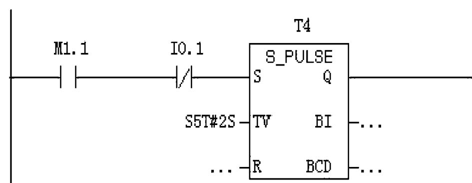

I0.4
I0.0
I0.7
I0.1
M10.0
M10.0

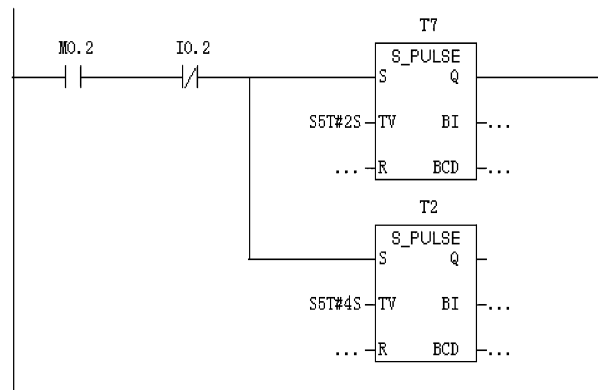

I1.0
I0.0
I0.7
I0.1
M10.1
M10.1


I0.5
I0.0
I0.2
M10.2
M10.2

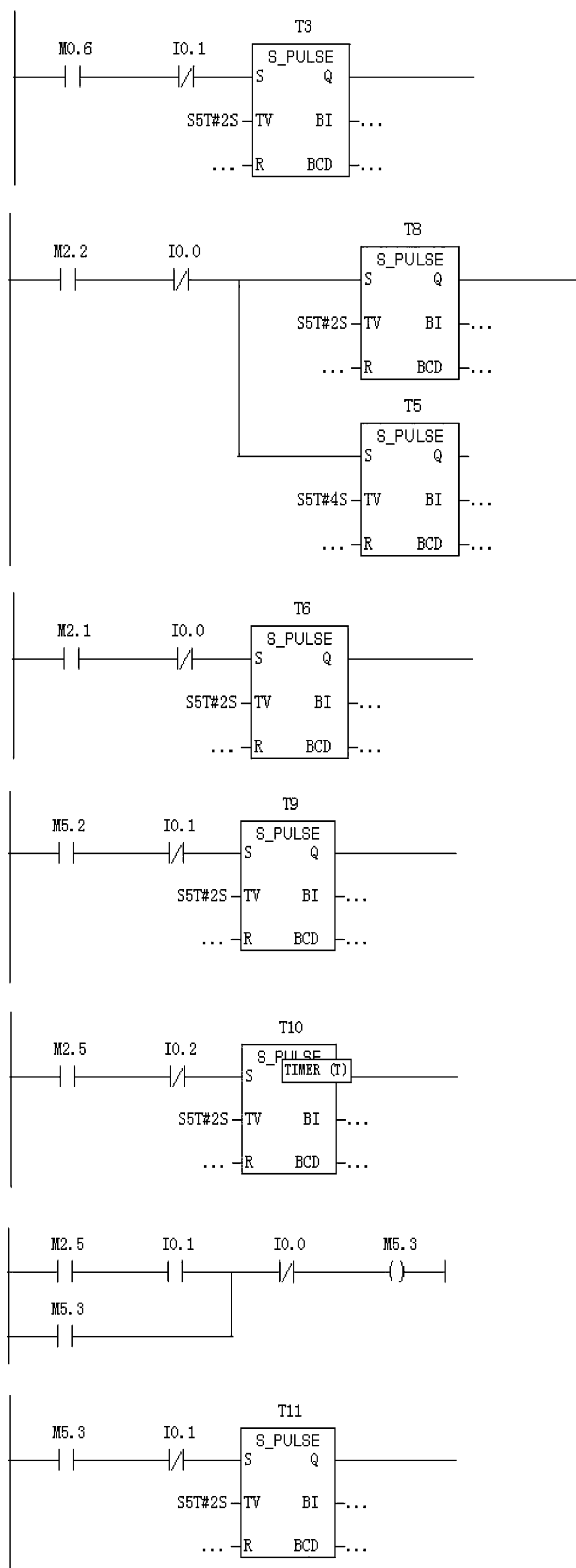

I1.1
I0.0
I0.2
M10.3
M10.3


T1
M0.4
I0.2
S_ODT
S
Q
S5T#2S
TV
BI
R
BCD


T4
M1.1
I0.1
S_PULSE
S
Q
S5T#2S
TV
BI
R
BCD

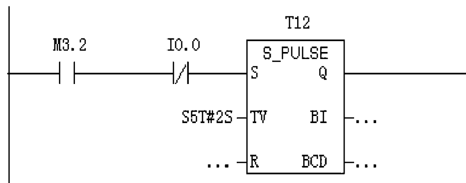

T7
M0.2
I0.2
S_PULSE
S
Q
S5T#2S
TV
BI
R
BCD
T2
S_PULSE
S
Q
S5T#4S
TV
BI
R
BCD

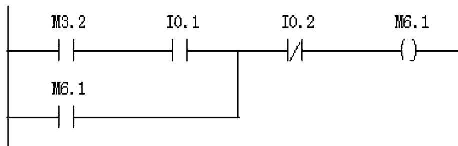
系统源程序（续）

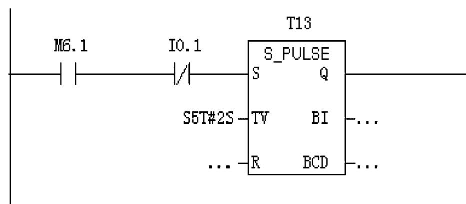


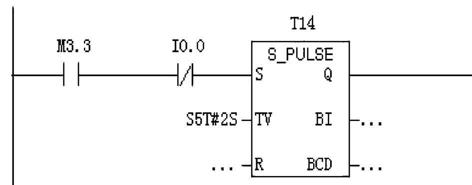
T3
M0.6
I0.1
S_PULSE
S
Q
S5T#2S
TV
BI
R
BCD
T8
M2.2
I0.0
S_PULSE
S5T#2S
T5
S_PULSE
S5T#4S
T6
M2.1
I0.0
S5T#2S
T9
M5.2
I0.1
S5T#2S
T10
M2.5
I0.2
TIMER (T)
S5T#2S
M2.5
I0.1
I0.0
M5.3
M5.3
T11
M5.3
I0.1
S_PULSE
S5T#2S

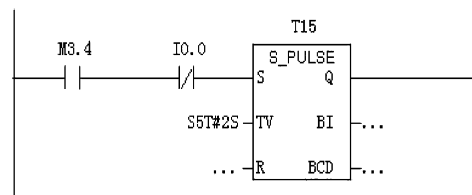
图 8-101　电梯控制

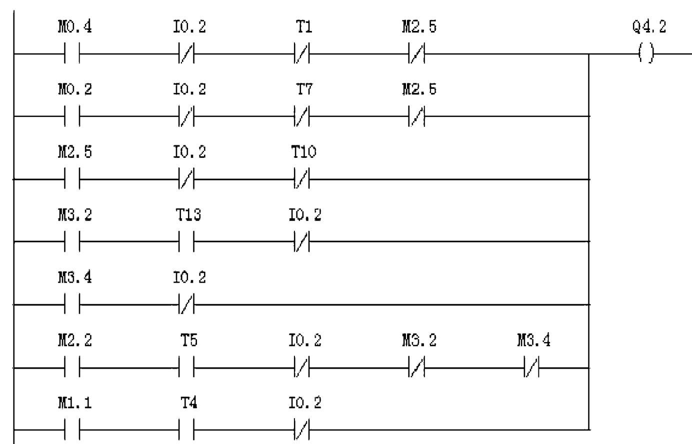

T12
M3.2
I0.0
S_PULSE
S
Q
S5T#2S
TV
BI
...
R
BCD


M3.2
I0.1
I0.2
M6.1
M6.1


T13
M6.1
I0.1
S_PULSE
S
Q
S5T#2S
TV
BI
...
R
BCD


T14
M3.3
I0.0
S_PULSE
S
Q
S5T#2S
TV
BI
...
R
BCD


T15
M3.4
I0.0
S_PULSE
S
Q
S5T#2S
TV
BI
...
R
BCD


M0.4
I0.2
T1
M2.5
Q4.2
M0.2
I0.2
T7
M2.5
M2.5
I0.2
T10
M3.2
T13
I0.2
M3.4
I0.2
M2.2
T5
I0.2
M3.2
M3.4
M1.1
T4
I0.2

系统源程序（续）

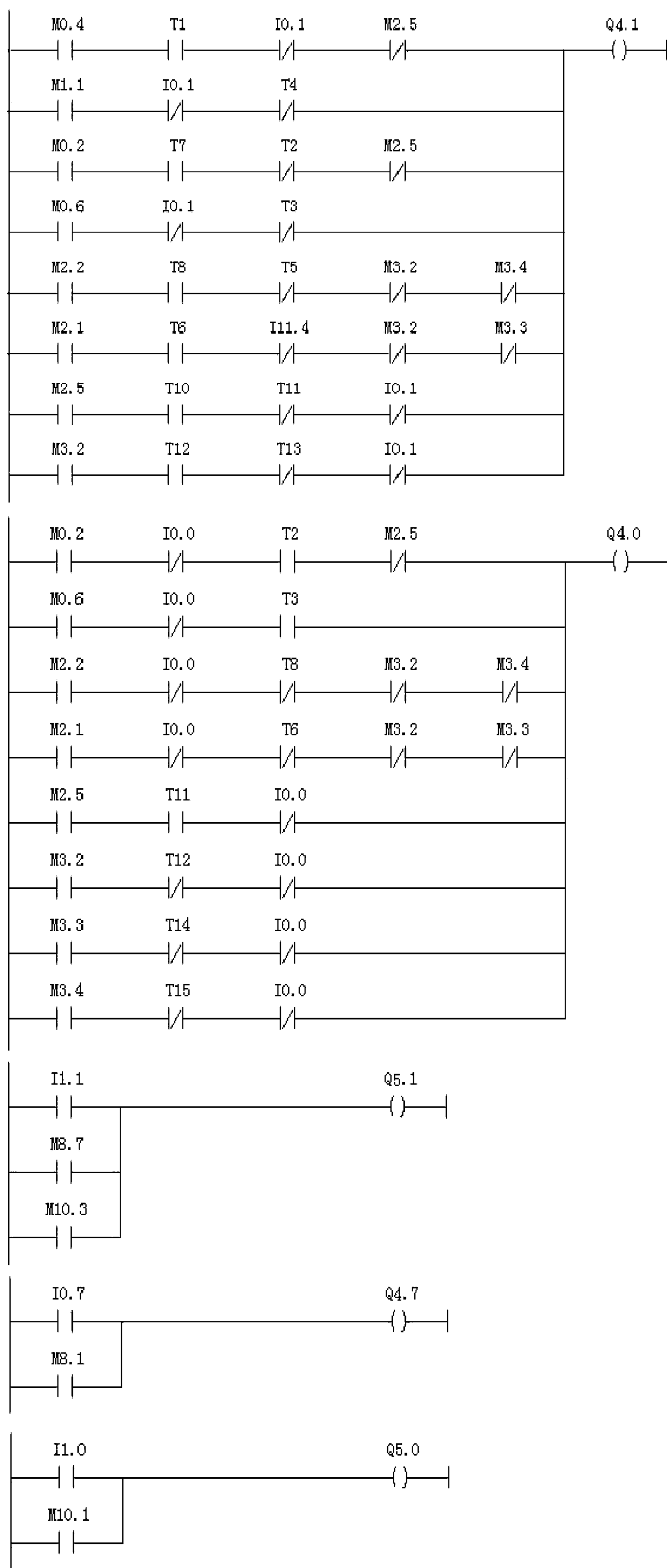


图 8-101 电梯控制系统源程序（续）

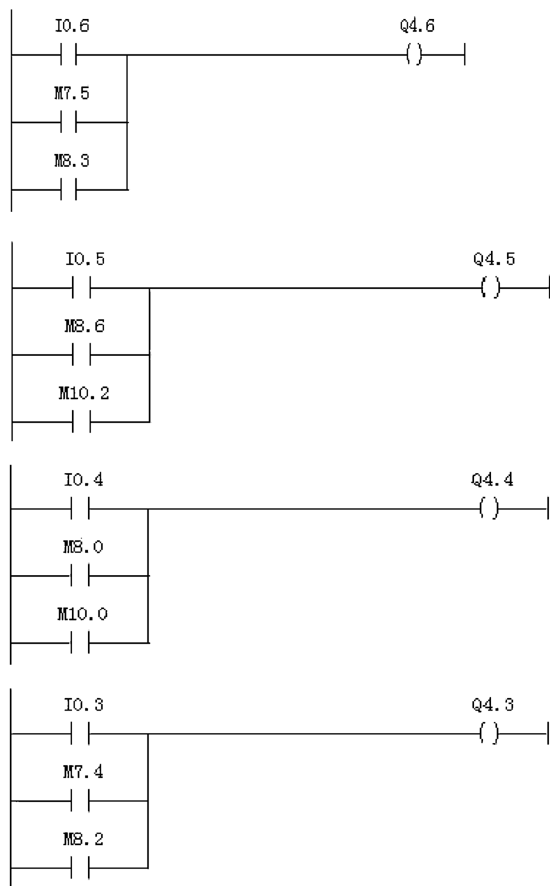


图 8-101　电梯控制系统源程序（续）

六、机械手控制系统线性程序设计

机械手控制系统的控制要求如图 8-102 所示。机械手一个循环周期可分为 10 步，分别为起动、下降、夹紧、上升、右移、下降、松开、上升、左移、停止。

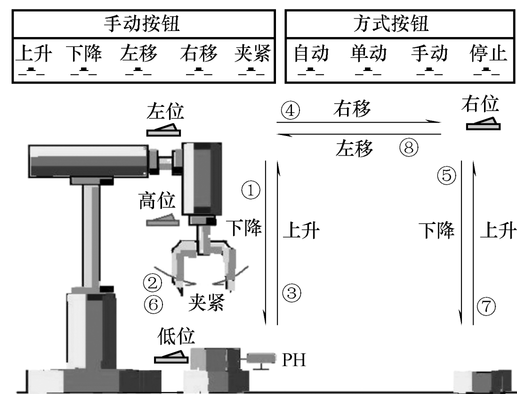


图 8-102　机械手控制系统的控制要求

控制方式：自动、单动和手动。

下面讨论自动控制过程。

1）I/O 地址分配见表 8-21。

表 8-21 I/O 地址分配表

模 块	输入端子号	输出端子号	地址编号	信号名称	描 述
CPU314	1		I0.0	自动起动，“1”有效	按钮
	2		I0.1	单次起动，“1”有效	按钮
	3		I0.2	手动起动，“1”有效	按钮
	4		I0.3	停止，“1”有效	按钮
	5		I0.4	高位，“1”有效	限位开关
	6		I0.5	低位，“1”有效	限位开关
	7		I0.6	左位，“1”有效	限位开关
	8		I0.7	右位，“1”有效	限位开关
	9		I1.0	手动下降，“1”有效	按钮
	10		I1.1	手动上升，“1”有效	按钮
	11		I1.2	手动夹紧，“1”有效	按钮
	12		I1.3	手动左移，“1”有效	按钮
	13		I1.4	手动右移，“1”有效	按钮
	14		I1.5	在工作，“1”有效	光耦合器
		1	Q0.0	下降，“1”有效	电磁阀
		2	Q0.1	上升，“1”有效	电磁阀
		3	Q0.2	右移，“1”有效	电磁阀
		4	Q0.3	左移，“1”有效	电磁阀
		5	Q0.4	夹紧，“1”有效	电磁阀

2）硬件接线原理图如图 8-103 所示。

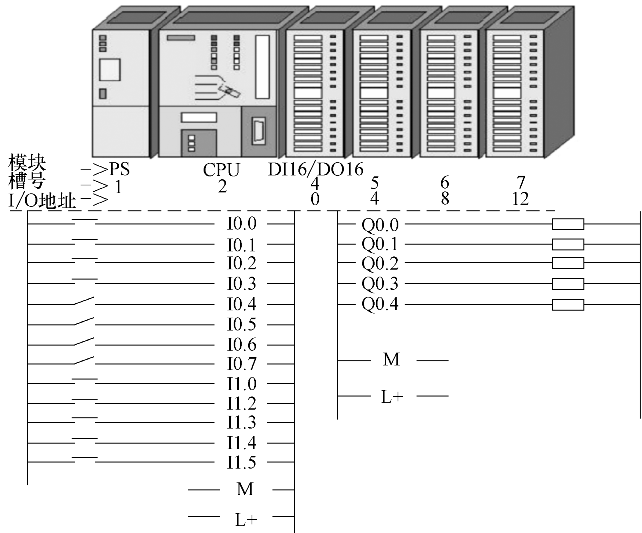


图 8-103 硬件接线原理图

3）由逻辑流程图设计程序，如图 8-104 所示。

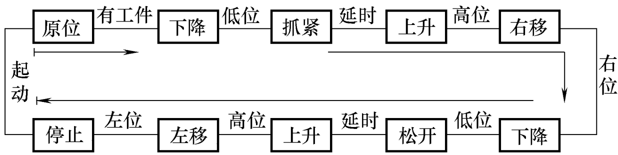


图 8-104 逻辑流程图

内存变量分配见表 8-22。

表 8-22　内存变量分配表

模　块　号	信号地址	信号名称	描　　述
1	I0.0	自动起动,"1"有效	按钮
2	I0.1	单次起动,"1"有效	按钮
3	I0.2	手动起动,"1"有效	按钮
4	I0.3	停止,"1"有效	按钮
5	I0.4	高位,"1"有效	限位开关
6	I0.5	低位,"1"有效	限位开关
7	I0.6	左位,"1"有效	限位开关
8	I0.7	右位,"1"有效	限位开关
9	I1.0	手动下降,"1"有效	按钮
10	I1.1	手动上升,"1"有效	按钮
11	I1.2	手动夹紧,"1"有效	按钮
12	I1.3	手动左移,"1"有效	按钮
13	I1.4	手动右移,"1"有效	按钮
14	I1.5	在工作,"1"有效	光耦合器
15	Q0.0	下降,"1"有效	电磁阀
16	Q0.1	上升,"1"有效	电磁阀
17	Q0.2	右移,"1"有效	电磁阀
18	Q0.3	左移,"1"有效	电磁阀
19	Q0.4	夹紧,"1"有效	电磁阀
20	T1	抓紧定时器	定时器
21	T2	放松定时器	定时器
22	M0.0	自动起动方式标志	标志位
23	M0.1	单次起动方式标志	标志位
24	M0.2	手动起动方式标志	标志位
25	M0.3	一周结束标志	标志位

4）由时序流程图设计程序。

由时序流程图设计程序，首先要把整个工程的各个任务分成多个时序，在不同的时序中完成不同的任务。本例中可分成 8 个时序。用 M1.0、M1.1…M1.7 分别表述各个时序的特征位。当 M1.0 = 1 时为机械手下降 1 时序，M1.1 为机械手抓紧时序等。时序流程图如图 8-105 所示。

5）线性结构软件设计如图 8-106 所示。

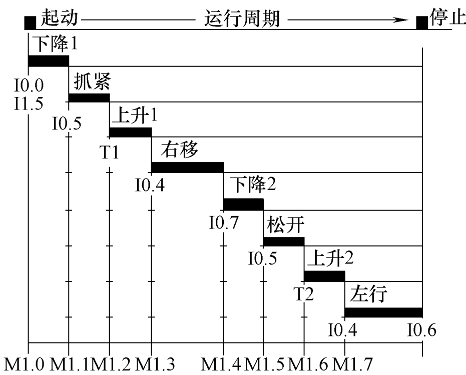


图 8-105　时序流程图

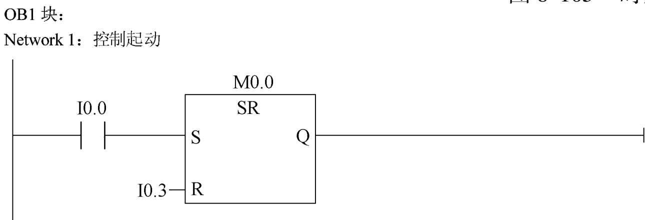
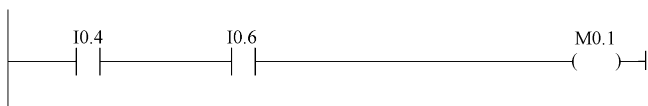
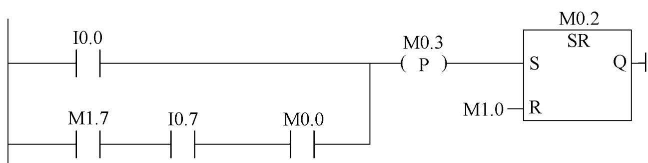


图 8-106　线性结构软件设计

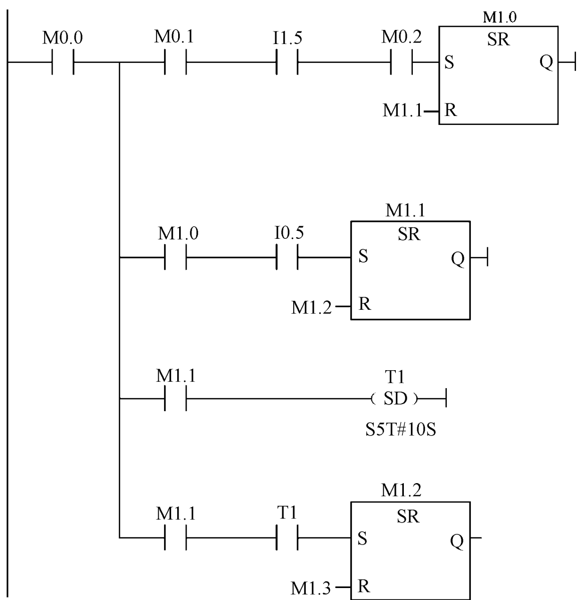
Network 2：设置单次起动方式



Network 3：下降



Network 4：旋转



Network 5：移动

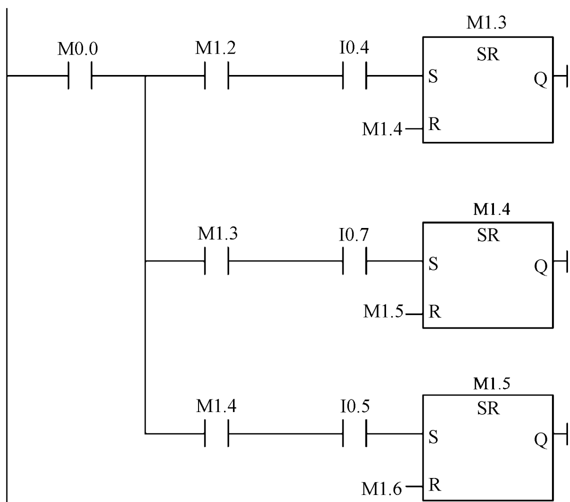
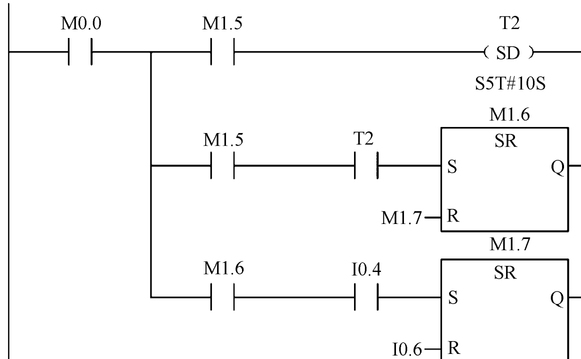
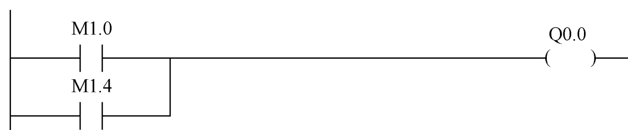


图 8-106 线性

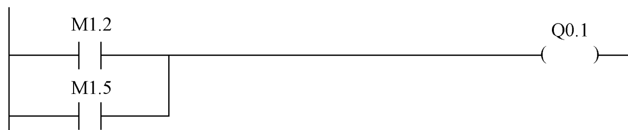
Network 6：移动停止



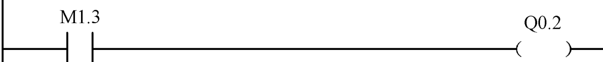
Network 7：输出电磁阀下降信号



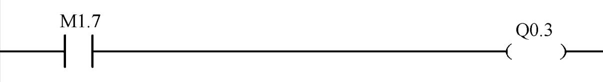
Network 8：输出电磁阀上升信号



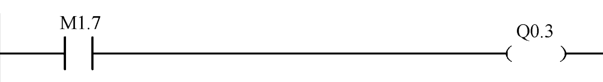
Network 9：输出电磁阀右移信号



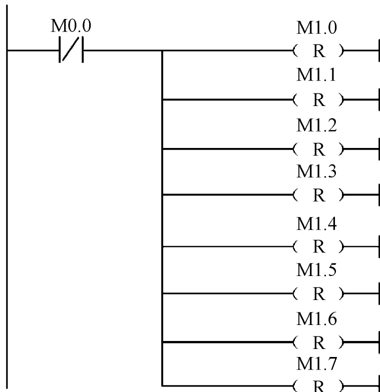
Network 10：输出电磁阀左移信号



Network 11：设置



Network 12：输出



结构软件设计（续）

第九章　常见故障现象与原因分析

第一节　常见故障的检查与处理

一、常见故障的总体检查与处理

总体检查的目的是找出故障点的大概范围，然后再逐步细化，确定具体故障点，达到消除故障的目的。常见故障的整体检查与处理的程序如图 9-1 所示。

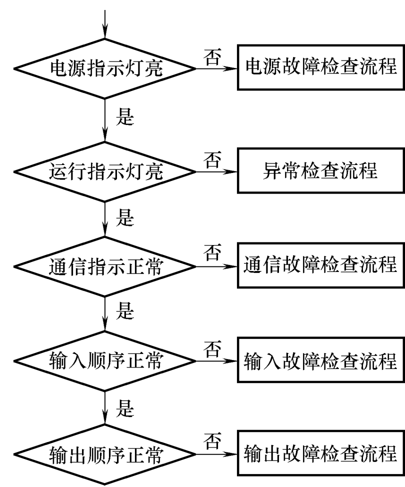


图 9-1　常见故障的整体检查与处理的程序图

二、电源故障检查与处理

PLC 系统主机电源、扩展机电源、模块中电源等显示不正常时都要进入电源故障检查流程，如果各部分功能正常，只能是 LED 显示有故障，否则应首先检查外部电源，如果外部电源无故障，再检查系统内部电源故障。电源故障检查与处理方法见表 9-1。

三、异常故障检查与处理

PLC 系统最常见的故障是停止运行（运行指示灯灭）、不能起动、工作无法进行，但是电源指示灯亮。这时，需要进行异常故障检查。异常故障检查与处理见表 9-2。

表 9-1　电源故障检查与处理方法

故 障 现 象	故 障 原 因	处 理 办 法
电源指示灯灭	指示灯坏或熔丝断	更换
	无供电电压	加入电源电压 检查电源接线和插座使之正常
	供电电压超限	调整电源电压在规定范围
	电源坏	更换

表 9-2　异常故障检查与处理

故 障 现 象	故 障 原 因	处 理 办 法
不能起动	供电电压超过上限	降压
	供电电压低于下限	升压
	内存自检系统出错	清内存、初始化
	CPU、内存板故障	更换
工作不稳定 频繁停机	供电电压接近上、下极限	调整电压
	主机系统模块接触不良	清理、重插
	CPU、内存板内元器件松动	清理、戴手套按压元器件
	CPU、内存板故障	更换

（续）

故障现象	故障原因	处理办法
与编程器(微机)不通信	通信电缆插接松动	按紧后,重新联机
	通信电缆故障	更换
	内存自检出错	内存清零,拔去电池,再连电池
	通信口参数不对	检查参数和开关,重新设定
	主机通信故障	更换
	编程器通信口故障	更换
程序不能装入	内存没有初始化	清内存,重写
	CPU、内存故障	更换

四、通信故障检查与处理

通信是 PLC 网络工作的基础。PLC 网络的主站、各从站的通信处理器、通信模块都有工作正常指示。当通信不正常时，需要进行通信故障检查。通信故障检查与处理见表 9-3。

表 9-3　通信故障检查与处理

故障现象	故障原因	处理办法
单一模块不通信	接插松动	按紧
	模块故障	更换
	组态错误	重新组态
从站不通信	分支通信电缆故障	拧紧插接件或更换
	通信处理器松动	拧紧
	通信处理器地址开关错	重新设置
	通信处理器故障	更换
主站不通信	通信电缆故障	排除故障、更换
	调制解调器故障	断电后再起动无效更换
	通信处理器故障	清理后再起动无效更换
通信正常,但通信故障灯亮	某模块插入或接触不良	插入并按紧

五、I/O 故障检查与处理

I/O 模块直接与外部设备相连，是容易出故障的部位，虽然 I/O 模块故障容易判断，更换快，但是必须查明原因，而且往往都是由于外部原因造成损坏，如果不及时查明故障原因，及时消除故障，对 PLC 系统危害很大。I/O 故障检查与处理见表 9-4。

表 9-4　I/O 故障检查与处理

故障现象	故障原因	处理办法
输入模块单点损坏	过电压,特别是串入高电压	消除过电压和串入的高电压
输入全部不通信	未加外部输入电源	接通电源
	外部输入电压过低	外加额定电源电压
	端子螺钉松动	将螺钉拧紧
	端子连接器接触不良	将端子锁紧或更换
输入全部断电	输入回路不通	更换模块
特定编号输入不通	输入器件不良	更换
	输入配线断线	检查输入配线
	端子接线螺钉松动	将螺钉拧紧
	端子连接器接触不良	将端子锁紧或更换

（续）

故障现象	故障原因	处理办法
特定编号输入不通	输入信号接通时间过短	调整输入器件
	输入回路不通	更换
	OUT 指令占用该输入号	修改程序
特定编号输出不关断	程序中输出指令继电器编号重复	修改程序
	输出继电器损坏	更换继电器
	漏电流或残余电压使其不能关断	更换负载
	输出回路不通	更换
输入不规则的通断	外部输入电压过低	使输入电压在额定的范围
	端子螺钉松动	将螺钉拧紧
	端子连接器接触不良	将端子锁紧或更换
异常输入点编号连续	输入模块公共端螺钉松动	将螺钉拧紧
	端子连接器接触不良	将端子锁紧或更换
	CPU 损坏	更换 CPU
异常输出点编号连续	输出模块公共端螺钉松动	将螺钉拧紧
	端子连接器接触不良	将端子锁紧或更换
	熔丝损坏	更换熔丝
输入动作指示灯不亮	指示灯损坏	更换指示灯

六、定期检修

定期检修的内容见表 9-5。

表 9-5　定期检修的内容

检修项目	检修内容	判断标准
供电电源	在电源端子处测量电压变化范围是否在标准范围内	电压变化范围：
		上限不高于 110% 供电电压
		下限不低于 85% 供电电压
外部环境	环境温度	0 ~ 55℃
	环境湿度	35% ~ 85% RH 不结露
	积尘情况	不积尘
输入输出用电源	在 I/O 端子处测电压变化是否在标准范围内	以各输入、输出规格为准
安装状态	各单元是否可靠固定	无松动
	电缆的连接器是否完全插紧，外部配线的螺钉是否松动	无松动
		无异常
寿命元件	电池、继电器、存储器等	以各元件规格为准
电源损坏	电源线引入过电压	把电源分析器连接到系统，检查过电压尖峰的幅值和持续时间，根据检查的结果给系统配置抑制设备
电子干扰问题	不合适的接地	纠正不正确的接地系统
	在控制柜内交叉配线	使控制盘良好接地，纠正高电压与低电压不合理的布线，把 DC 24V 传感器电源的 M 端子接地
	对快速信号配置了输入滤波器	增加系统数据块中的输入滤波器的延迟时间

（续）

检修项目	检修内容	判断标准
当连接一个外部设备时通信网络损坏 （计算机接口、PLC 的接口或 PC/PPI 电缆损坏）	如果所有的非隔离设备（例如 PLC、计算机的其他设备）连到一个网络，而该网络没有一个共同的参考点，通信电缆提供了一个不期望的电流通路。这些不期望的电流可以造成通信错误或损坏电路	检查通信网络
		更换隔离型 PC/PPI 电缆
		当连接没有共同电气参考点的机器时使用隔离型 RS-485 to RS-485 中继器

七、PLC 的故障处理

对于具体的 PLC 的故障检查可能有一定的特殊性。下面给出了有关 PLC 的故障检查与处理方法，见表 9-6。

表 9-6　PLC 的故障检查与处理方法

问　　题	可能原因	处理方法
输出不工作	被控制的设备产生了损坏输出的电气浪涌	当接到感性负载时，需要接入抑制电路
	程序错误	修改程序
	接线松动或不正确	检查接线，如果不正确，要改正
	输出过载	检查输出的负载
	输出被强制	检查 CPU 是否有被强制的 I/O
CPU SF（系统故障）灯亮	用户程序错误	对于编程错误，检查 FOR、NEXT、JMP 和比较指令的用法
	看门狗错误	
	间接寻址	
	非法的浮点数	排除电气干扰
	电气干扰	检查接线、控制盘良好接地和高电压
		与低电压没有并行引线是很重要的
		把 DC 24V 传感器电源的 M 端子接地
	元件损坏	更换元件

第二节　常见问题及解答

一、如何将二线制测量传感器连接到模拟量模块、紧凑型 CPU 或 C7 设备

解答：关于模拟量模块配置二线制测量传感器请参阅相关手册的描述。

此描述包括以下配置：

1）只能配置四线制测量传感器的模拟量模块。

2）在一个通道组上的二线制和四线制测量传感器并存。

3）紧凑型 CPU 或 C7 设备。

不同测量传感器的定义：

1）四线制测量传感器由单独的电源供电，两条测量电缆连接到模拟模块的 M + 和 M – 。由于这个原因它们也称为有源传感器。

2）二线制测量传感器也叫做无源传感器，因为它们通常通过模拟量模块供电。

要将二线制测量传感器连接到标准模拟量模块、紧凑型 CPU 或 C7 设备，需要一个外部 24V 电源（见图 9-2）。此二线制测量传感器的电源必须是防短路的。使用熔断器来保护电源单元。请注意如果在测量传感器发生故障（短路）时模拟输入端未被保护。

注意事项：

1）输入标号取决于模块类型；例如 M＋、M－。

请参阅单个模块的电路图。

2）在硬件配置中模拟模块必须按照四线制测量传感器进行配置。

这也同样应用于在一个通道组上同时连接二线制和四线制传感器的情况。

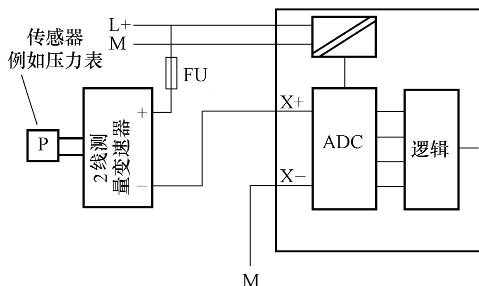


图 9-2 连接二线制测量传感器

X＋—测量输入＋ X－—测量输入－

M—接地端子 L＋—DC 24V 电源端子

二、S7-300 模拟量输入模块测量温度时的测量误差

问题：

用 S7-300 模拟量输入模块测量温度（华氏）时，可以使用模块文档中列出的误差极限吗？

解答：

不可以直接使用指定的误差极限。基本误差和运行误差都以绝对温度或摄氏温度说明。必须乘以系数 1.8 将其转换为华氏温度单位。

例：

S7-300 AI 8×RTD：对热电阻输入模块指定的操作误差是＋/－1.0 摄氏度。

当以华氏温度测量时，可接受的最大误差是＋/－1.8 华氏度。

三、把一个 PT 100 温度传感器连接到 SM331

问题：

如何把一个 PT100 温度传感器连接到模拟输入模块 SM331？

解答：

PT100 热电阻随温度的不同其电阻值随之变化。如果有一恒定电流流经该热电阻，该热电阻上的压降随温度而变化。恒定电流加在接点 Ic＋和 Ic－上。模拟模块 SM331 就可通过 M＋和 M－点测定出电流的变化。

通过测定电压就可以确定出温度。PT100 到模拟输入组有三种连接方法（见图 9-3）：4 线连接可得到最精确的测定值。

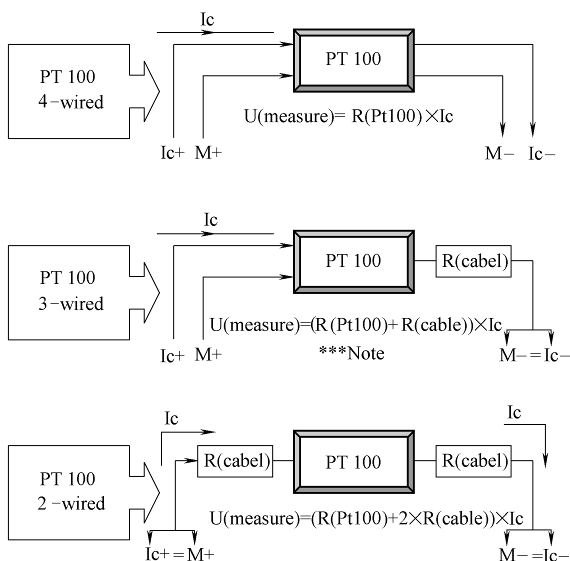


图 9-3 PT100 到模拟输入组的三种连接方法

注意：

对于 3 线连接用的公式仅表明了模拟量输入模块 SM331（MLFB 为 6ES7 331-7Kxxx-0AB0）的实际测定过程。

在 S7-300 系列中，存在一些通过多次测定的模拟输入端。它们规定出公共返回线的线电阻并做数学补偿。所获精确度几乎可与 4 线连接媲美。这种模块的一个例子就是 SM331（MLFB 为 6ES7 331-7PF00-0AB0 和 6ES7 331-1KF01-0AB0）。

此时，上述公式只作为物理模型的原理，并不能反映出实际的 PT100 的测量关系。

四、将 HART 测量传感器连接到常规的 S7-300 模拟输入模块是可行的

问题：

可以将 HART 测量传感器连接到 SIMATIC S7-300 系列常规的模拟量输入模块吗？

解答：

如果不需要 HART 测量传感器的任何 HART 特性，就可以使用其他 S7-300 模拟输入模块。例如，可以使用模块 6ES7 331-7KF0x-0AB0 或一个带隔离的 4 通道模块（如 6ES7 331-7RD00-0AB0）。为此，要将积分时间设置为 16.66ms、20ms 或 100ms。对于连接到手持式设备，或与手持式设备通信，电路中必须串接一个 250Ω 的电阻。

注意事项：

如果要通过控制器（比如说，SIMATIC PDM）来编程 HART 测量传感器，必须使用一个相应的 HART 模块（例如，6ES7 331-7TB00-0AB0 或 6ES7 332-5TB00-0AB0）。

五、有关 SM 335 正确接线的信息

问题：

如何避免 SM335 模块中模拟输入的抖动？

解答：

下列接线（见图 9-4）适于下列 MLFB 的模拟量 I/O 模块：

- 6ES7335-7HG00-0AB0
- 6ES7335-7HG01-0AB0

检查安装的传感器是否接地或与地隔离安装。

安装在绝缘机架上的传感器：将接地端子 M_{ana}（针 6）连接到测量通道 M_0（针 10）、M_1（针 12）、M_2（针 14）和 M_3（针 16）以及中央接地点（CGP）时，尽可能通过最短路径（可能的话，直接连接到前连接器）。

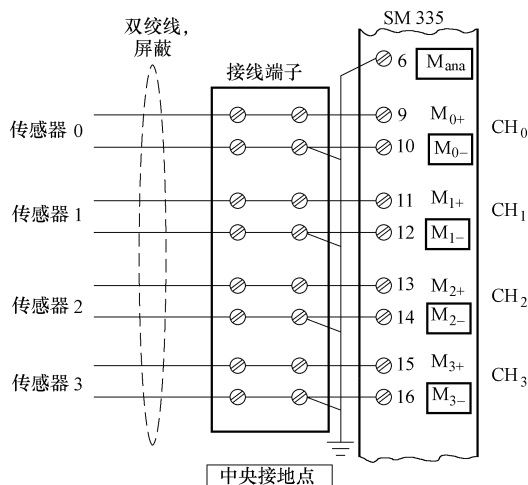


图 9-4　模拟输入的接线

接地传感器：

为了确保传感器有良好的等电位连接。最好是把从 M 到 M_{ana} 和到中央接地点的连接隔离起来。请将屏蔽层两端接地。

这种做法可能会导致测量中存在干扰，最好使用隔离的传感器。

六、怎样理解 S7-400 数据的存储及存储容量，如何查找 CPU 的存储器参数

装载内存：

在 CPU 的装载内存中可以装载所有块，并包括块参数接口所占用的地址空间，也可以归档数据块，比如，通过调用系统功能将数据块只存储于装载内存中。

在 S7-400 CPU 中，可以插入的外部存储卡如 FLASH 闪存卡或 RAM 卡扩展 CPU 的装载内存。

工作内存：

工作内存只存储与程序顺序调用相关的数据。存储器的一半用于存储程序，另一半存储数据（这是一个固定分配）。

工作内存集成在 CPU 上且不能扩展。如果工作内存对于一个应用程序来说不够大，则必须更换更大内存的 CPU。

工作内存通过电池备份。

系统内存

系统内存包括以下存储区域：

- 过程映像输出和过程映像输入（PIQ，PII）
- 标志器（M）
- 定时器（T）
- 计数器（Z）
- 本地数据栈（L 栈）

使用存储卡扩展装载内存：

当使用 RAM 存储卡时，必须使用电池对存储卡上的数据以及内部 RAM 上的数据进行备份，避免系统掉电而造成数据丢失。

当使用闪存 FLASH 存储卡时，因为 FLASH 是非易失性的存储器，用户程序不需要电池备份，但是 CPU 集成 RAM 上的过程数据需要电池备份。

当新插入内存卡时，操作系统请求完全复位（STOP LED 慢闪烁）。如果模式选择器保持在“MRES”位置，系统将执行一次完全复位，之后用户程序从存储卡传送到 CPU 的工作内存中。

重要事项：

程序运行时不能插拔存储卡。

上电之后 CPU 的动作：

CPU 上电后可以自动识别是否有电池备份。

如果 CPU 上电有电池备份，则工作内存的程序用于进行下一步操作。

如果 CPU 上电没有电池备份，则用户程序从装载内存传送到工作内存。

重要事项：

CPU 318-2DP 的存储原则与 S7 400 CPU 系列相似。

在 SIMATIC Manager 上先选择站点中的 CPU，然后点击菜单“PLC→Module information...”并选择“Memory”标签，可以查看 CPU 存储器参数。

七、如何利用 OB81 判断电源故障

如：电池故障

解答：

如果电源（仅 S7-400）或缓冲区中的一个错误触发一个事件，则 CPU 将调用 OB81。错误纠正后，重新调用 OB81。在电池故障的情况下，如果电池检测中的 BATT. INDIC 开关是激活的，则 S7-400 仅调用 OB81。如果没有调用 OB81，则 CPU 不会进入 STOP 状态。如果 OB81 不可用，则当电源出错时，CPU 仍保持运行。

在样例程序（见图 9-5）中，使用 OB81 的临时变量“OB81_FLT_ID”判断电池故障，这种情况下，变量包含出错代码“ 22_{hex} ”。如果电池故障，则置位 M81.1。通过变量“OB81_EV_CLASS”s，可以识别两种类型的事件：

```
OB81 :  "Power Supply Fault"
Network 1: Error of Battery, incoming event
        L      #OB81_FLT_ID        //Load the error code
        L      B#16#22             //Error code of missing voltage
        ==I
        =      M      81.1         //Set flag for error of b
```

图 9-5 OB81 中出错评估的样例程序

- B#16#39：新来事件，电池故障；
- B#16#38：退出事件，电池故障已清除。

通过 M81.0 标志位判断故障的有无。

如果标志位 M81.1 和标志位 M81.2 被置位（电源故障和新来事件），则图 9-5 中（OB81 中的样例程序）标志位 M81.0 置位，标志 M81.0 复位为退出事件。

注意事项：

上述程序只有当 CPU 实际运行时出现电池故障才有效。如果 CPU 在 STOP 模式时出现电池故障，则新来事件（调用 OB81）只在 CPU 重新回到运行模式时才被触发。如果关闭电源，则不会触发新来事件（电池故障）。

八、如何能在不重新起动系统的情况下，改变 PUT 和 GET SFBs（SFB14、15）**的 ID 参数**

解答：

ID 参数包含了 S7 连接的号。如需要改变，必须初始化系统功能块。而初始化过程在起动 CPU 时就已完成。不过，通过重新下载 SFB 背景数据块，可以在不将 CPU 切换为 STOP 模式的情况下改变该参数。但重要的是，在这种情况下数据块里包含了初始调用给出的初始值。如果不需要每次调用都给出该参数，而是在背景数据块预先分配该参数，可以通过块调用赋值 ID 参数。

如果背景 DB 没有初始化，选择命令“数据浏览→编辑→初始化数据块”将它初始化。

九、使用系统功能块 SFB12 和 SFB13（BSEND BRCV）**时应注意些什么**

解答：

使用系统功能块 SFB12 和 SFB13（BSEND/BRCV）时，必须遵守以下几点。

有关使用系统功能块 SFB12 的说明：

1）REQ 参数通过输入的上升沿起动一个任务。在 DONE 或 ERROR 位还没有置位前，该作业一定不能复位和重起动。在每发送一个作业后相应会置位这两个位中的一个。之后 REQ 参数就可以再次触发下一个发送任务。为确保功能性，在系统功能块的 REQ 输入处至少要有一个上升沿。

2）如果 ERROR 位置位，需要判断参数“STATUS”，以便能检索相关出错的详细信息，从而能直接清除它。有必要的话，可对此“STATUS”做一般的判断。因为有一种情况（STATUS CODE：11）下，ERROR 位不一定被置位。

3）当连接建立后，“ID”参数包含了连接参数。

4）在相互连接的两个系统（站）中，“R-ID”必须是相同的，并且在系统中是唯一的。

5）只有在首次调用系统功能块时，参数“SD_1”（ANY 类型）的长度才被评估，并且依据它的值建立发送缓冲区。该值规定了通信数据的最大量。后续的调用中，只评估 LEN 参数并依据参数定义的数据量传输数据。

“ID”和“R_ID”不可动态赋值，因为它们只有在首次调用时被评估和设置。它们不可在后续的调用里被更改。

有关使用系统功能块 SFB13 的说明：

1）参数“EN_R”可永远为 1。因为此系统功能块异步工作。

2）对于参数“ID”和“R_ID”，用于系统功能块 SFB12 的第 3 点和第 4 点同样适用于它。

3）对于参数“RD_1”用于系统功能块 SFB12 的第 5 点同样适用于它。

4）上述通信过程完成后，不是“NDR”就是“ERROR”被置位。只有在 NDR 位已经置位

后，才能访问接收缓冲区里的数据（保证数据完整性）。

5）只有当 ERROR 位已置位，才能评价“STATUS”，就像系统功能块 SFB 12 的第 2 点里所述的那样。

注意：

请确保“DONE”、“NDR”、“ERROR”和“STATUS”里的数据只在事件出现的那个周期里可用。相应地，评估和事件处理也同时发生，或至少在同一个周期里触发。

十、为什么具有诊断功能的数字输出模块 SM422-7BL 的外部故障灯（EXTF），在清除输出与地短路的情况下仍常亮

解答：

这个问题的原因在于该模块的错误检测机制：如果模块的输出出现一个对地短路或者其中一个输出端过载，那么该模块只有在有效输出信号时才会探测这个错误，也就是说诊断警报 OB（OB82，进入）的调用以及 EXTF 发光二极管的激活，只有在输出信号从“0”到“1”转变时才会发生。同样，短路或过载的消除也只有当输出激活时才被探测。诊断 OB（输出）的调用和 EXTF LED 的取消激活也只在错误消除后才会进行。

十一、当测量值为“7FFF”时，如何分辨是断线故障还是测量值溢出

解答：

通过参数化的产生诊断中断的原因能有效判断产生“7FFF”测量值的原因，设置断线产生诊断中断，测量值溢出不产生诊断中断，这样：

- “测量值 7FFF 并产生诊断中断意味着发生了断线故障”；
- “测量值 7FFF 不产生诊断中断意味着测量值溢出”。

在 OB82 中，可以判断出产生诊断中断的信息（比如：模块、通道），并且定义对该中断所做出的响应。可以使用系统功能块 SFC51“RDSYSST”将数据从相关的系统状态列表“SSL”中读出。

十二、为什么尽管插入一块新的备用电池还出现电池故障信号

问题描述：

S7-400CPU 使用的是锂电池（锂/亚硫酰氯）。锂电池在长期放置的情况下会生成钝化膜，这个特性会直接影响备用电池的功能。这种情况下，当系统电源接通时，CPU 会提示错误信息。后备电池在电源模块通电的情况下会被去钝化，将锂电池放入 S7-400 的电源单元清除锂电池的钝化膜。这个过程需要几分钟的时间。当钝化膜被清除且锂电池达到其额定电压时，可以使用 FMR 按钮来确认电源模块的错误信息。

由于锂电池的存储时间通常不可推测，推荐以下步骤去除钝化膜：

1）在电池舱中放入备用电池（组）。

2）用 FMR 按钮确认电源单元的所有电池故障信息。

3）如果无法确认电池故障，等几分钟后再试。

4）如果依然无法确认电池故障，拔出电池（组）并将它们短路 1～3s，注意短路时间不要超过 3s。

5）重新插入电池（组）并用 FMR 按钮再次尝试确认。

6）当所显示的电池故障信息消失后，电池（组）可以准备使用。

7）如果所显示的电池故障信息不消失，则电池（组）没电。

十三、何时更换 S7-300/400 控制器的备用电池

问题描述：

推荐使用以下做法确定何时更换控制器的备用电池：

- 每年更换一次 S7-300 控制器的备用电池。

注意事项：

带有 MMC 卡的 CPU 不需要备用电池或备用时钟电池。

- 计算 S7-400 控制器的缓冲时间并根据该时间更换备用电池。

注意事项：

在运行中不能更换备用时钟电池，备用时钟电池只能在 S7-300 CPU 中使用。如果使用的是备用时钟电池而不是备用电池，那么如果发生电源故障，只有硬件时钟的时间日期值会被保存，此时控制器的“BATF”指示灯亮。

重要事项：

必须在“Power ON”的模式下来更换备用电池，否则用户内存中的数据会丢失且 CPU 内部时钟会停止。

十四、当采用交流电源供电时，应该选择哪种电源进线断路器

解答：

当把 PS407 电源接到交流电源上时，推荐使用 10A 额定电流且具有 C 触发特性的断路器。

十五、当 24V 电源过载时 S7-400 有何反应，电源模块又如何反应

解答：

对于 S7 400 的所有电源模块，如果 24V 电源过载，输出电流就会被限制在额定值的 100% ~ 150% 之间。表 9-7 中给出的是各种电源模块的输出电流的额定值。

表 9-7　各种电源模块的输出电流的额定值

电源模块	额定值	电源模块	额定值	电源模块	额定值
6ES7 405-0DA00-0AA0	0.5A	6ES7 405-0KA01-0AA0	1.0A	6ES7 407-0KR00-0AA0	1.0A
6ES7 405-0DA01-0AA0	0.5A	6ES7 405-0KR00-0AA0	1.0A	6ES7 407-0RA00-0AA0	1.0A
6ES7 407-0DA00-0AA0	0.5A	6ES7 405-0RA00-0AA0	1.0A	6ES7 407-0RA01-0AA0	1.0A
6ES7 407-0DA01-0AA0	0.5A	6ES7 405-0RA01-0AA0	1.0A		
6ES7 405-0KA00-0AA0	1.0A	6ES7 407-0KA01-0AA0	1.0A		

如果 24V 电源掉到了低电压门槛值（允许 0% ~ +5%，相应为 19.2 ~ 20.16V）19.2V 以下，该模块触发总线信号“/PF24”（电源电压低）。CPU 的“EXTF”LED 指示灯（外部错误）点亮，并把该错误保存到诊断缓冲区里。此时 OB81 被调用。4A 型电源模块（额定电流为 0.5A）的其他反应如下：

1）24V 电源临时关断。

2）经过大约 0.5 ~ 1s 的间隔，24V 电源再次打开。一旦电源再次达到额定范围，也就是说超过低电压门槛，模块进入正常工作模式。

10A 和 20A 型电源模块（标称限制电流 1.0A）过载时的其他反应如下：

1）电压自动按负载电阻调整，所以额定电流（1.0A 到最大 1.5A）不会超限。电源模块进入受控工作模式。

2）一旦过载消失，电源回到额定范围，绿色“24V”LED 指示灯闪动。

十六、300 系列以太网 CP 模板有什么不同

组态注意事项：

300 系列中的工业以太网 CP 模板在其硬件接口和通信功能包括数量框架在内有所不同。

表 9-8 显示了区别。

表 9-8　区别

通信处理器	CP343-1	CP343-1	CP343-1 IT	CP343-1 PN	CP343-1 Lean
MLFB	6GK7 343-1EX11-0XE0	6GK7 343-1EX21-0XE0	6KG7 343-1GX21-0XE0	6GK7 343-1HX00-0XE0	6GK7 343-1CX00-0XE0
连接	ITP，RJ45，AUI	RJ45	RJ45	ITP，RJ45，AUI	RJ45
应用					
ISO 传输协议	×	×[1)]	×[2)]	—	—
ISO-on-TCP 传输协议	×	×	×	×	×
TCP/IP 传输协议	×	×	×	×	×
UDP 传输协议	×	×	×	×	×
S7 通信	×	×	×	×	×[5)]
IT 通信	—	—	—	—	—
PG/OP 通信	×	×	×	×	×
PROFINET 通信	—	×	×	×[4)]	—
S7 路由	×	×	×	×	×
数量框架					
S5-兼容通信[3)]	16	16	16	16	16
用户数据量 TCP	8K 字节	8K 字节	8K 字节	8K 字节	8K 字节
用户数据量 UDP	2K 字节	2K 字节	2K 字节	2K 字节	2K 字节
S7 通信数	16	16	16	16	4
PG/OP 通信数（非循环效用）	16	16	16	16	4
多协议（同时操作的所有连接总数）	48	48	48	32	12
组播	16	16	16	16	8

注：1. 早期的 CP343-1EX20 模块既不支持 ISO 协议也不支持 PROFINET。
2. 早期的 CP343-1GX20 模块既不支持 ISO 协议也不支持 PROFINET。
3. 同时操作的所有 TCP/UDP 连接总数。
4. 只支持 PROFINET CBA 版本 V1.0（非实时）。
5. 只支持服务器功能。

区别标准的描述：

ISO 传输协议：ISO 传输连接用于 S5 站和 S7 站或 PC 站之间的数据交换（S5 兼容通信）。

ISO 传输连接的属性：站间的通信是基于 MAC 地址的。使用数据块的数据传输适用于大量数据。可使用“SEND/RECEIVE”和“FETCH/WRITE”应用实现数据传输。在 PC 上，ISO 传输服务通过 C 函数或利用 OPC 服务器提供。数据的接收是由对方通过 ISO 参考模型第 4 层上的确认来确定的。数据不能通过路由器（非路由功能的协议）传递。

ISO-on-TCP 传输协议：ISO-on-TCP 传输连接用来进行 S5 站和 S7 站或 PC 站间的数据交换（兼容 S5 的通信）。

ISO-on-TCP 连接的属性：站间的通信是基于 IP 地址的。符合 TCP/IP 标准的 FRC 1006 扩展是与 ISO 参考模型的第 4 层相一致的。使用“SEND/RECEIVE”和“FETCH/WRITE”实现数据传输。在 PC 上，ISO 传输服务通过 C 函数或利用 OPC 服务器提供。数据的接收是由对方通过 ISO 参考模型第 4 层上的确认来确定的。数据可通过路由器（有路由功能的协

议）传递。

TCP/IP 传输协议：通过 TCP 连接的配置实现站间（包括第三方的站）的数据交换。

TCP 连接的属性：符合 TCP/IP 标准。可使用“SEND/RECEIVE”和“FETCH/WRITE”实现数据传输。操作系统中已存在的 TCP/IP 通常可用在 PC 上。数据可通过路由器（有路由功能的协议）传递。

UDP 传输协议：通过 UDP 连接的配置实现两站间的数据交换。

UDP 连接的属性：UDP 协议两站之间关联数据块的不可靠传输，支持组播传输通过建立组播传输环，组播传输允许站组一起接收信息和发送信息到这个组。通过“SEND/RECEIVE”服务进行数据传输。数据可通过路由器（有路由功能的协议）传递。

S7 通信：通过 S7 连接的配置实现 S7 站间和 PC 站间的数据交换。

S7 连接的属性：该连接可用于所有 S7/M7 设备。可用于所有子网（MPI、PROFIBUS、工业以太网）。SIMATIC S7/M7-300/400 站之间数据的可靠传输（使用“BSEND/BRCV”或“PUT/GET”SFB）。高速、不可靠数据传输取决于对方与时间相关的操作（使用“USEND/URECV”SFB）。在 ISO 参考过程的第 7 层上确认对方的数据传输。

IT 通信：

E-mail 功能：S7 站激活报警并将其文本以电子邮件发送。电子邮件可能包含变量值和附件。

HTML 功能：CP 模板中装载了 Web 服务器。其他的 Applets 或 Java beans 同样可用于提供和查看带有 S7 变量的 HTML 页。JAVA 编写的应用程序可通过 Java beans 使用 HTTP 协议访问 S7 变量。

FTP 功能（作为服务器和客户端）：FTP 客户端功能可用于在 FTP 服务器上将数据（仅来自于数据块）保存为文件或从 FTP 服务器上取回数据（仅在带有特殊首部（20 字节）的块中）。FTP 服务器功能用来将文件（HTML 页，映像文件，......）传输到 CP 的文件系统。也可直接从数据块中读出值或通过文件直接把值写入数据块中。

PG/OP 通信：通过以太网用 STEP 7 编程和组态 S7 站。编程设备是连接到以太网的。

PROFINET 通信：PROFINET 是 PROFIBUS 用户组织（PNO）使用的标准，它定义了跨厂商通信和工程模型。它用在基于组件的自动化（CBA）环境中。

十七、SIMATIC S7-300/400 如何使用 BSEND BRCV 确保数据传输的一致性

使用 BSEND/BRCV 最多可传输以下组的数据：

1）通过 MPI、PROFIBUS 传输 32K 字节。

2）通过以太网传输 64K 字节。

一致性传输是以块为单位。一个块包含：

1）在 SIMATIC S7-400 中 440 字节的用户数据。

2）在 SIMATIC S7-300 中 220 字节的用户数据。

用户不必考虑指定块的大小。用户可通过正确分析 BSEND 块的 DONE 参数和 BRCV 块的 NDR/LEN 参数来保证数据的一致。

BSEND：为了保证数据的一致性，在当前传送过程没有完成之前不要再次写入正在使用的 SD_1 发送区域部分。这是当 DONE 状态参数已变为 1 时的情况。

BRCV：为了保证数据的一致性，在接收过程完成之前不要分析 RD_1 的接收区域。这是 NDR 状态参数已变为 1 时的情况。建议通过 LEN 状态参数来确定所用 RD_1 接收区域的长度。

十八、哪些 IP 地址与哪些子网掩码相互兼容

组态注意事项：

关于 IP 地址域和所谓的“默认子网掩码”分配有个惯例。利用 IP 地址的第一个十进制数（左起）中数值“1”（二进制）的数量定义默认子网掩码结构如下：

IP 地址（十进制）	IP 地址（二进制）	地址类别	默认子网掩码
1 to 126	0xxxxxxx. xxxxxxxx...	A	255.0.0.0
128 to 191	10xxxxxx. xxxxxxxx...	B	255.255.0.0
192 to 223	110xxxxx. xxxxxxxx...	C	255.255.255.0

注意事项：

对于 IP 地址的第一个十进制数也可使其值在 224 和 225 之间（地址类别 D 等）。但是，不推荐这样做，因为 STEP 7 的确会对这些值做地址检查。

通过子网掩码可进一步构建一个分配到一个地址类别 A、B 或 C 的子网，并通过设置子网掩码的其他低值点为“1”建立“专用”的子网。每设置一位为“1”，“专用”网络的数量就加倍，其中的节点数量减半。表面上看网络仍然像一个单一网络。

样例：

在 B 类地址子网中（例如 IP 地址 129.80.xxx.xxx）可改变默认子网掩码如下：

掩码	十进制	二进制
默认子网掩码	255.255.0.0	11111111.11111111.00000000.00000000
子网掩码	255.255.128.0	11111111.11111111.10000000.00000000

所有节点地址从 129.80.001.xxx ~ 129.80.127.xxx 的节点都位于一个子网上，所有节点地址从 129.80.128.xxx ~ 129.80.255.xxx 的节点位于另一个子网上。

注意事项：

更多信息可参见 STEP 7 的在线帮助。

请注意当建立连接时所有的伙伴必须在同一子网内或确保正确的子网路由。

十九、如何在 TCP/IP 网络中分配 IP 地址和子网掩码

问题：

在 TCP/IP 网络中分配 IP 地址和子网掩码需要注意些什么？

解答：

在网际协议（IP）中规定，每个已连接的节点需分配一个唯一标识的 IP 地址。根据 IP 协议版本 4（IPv4），一个 IP 地址由 4 个部分组成，这 4 个部分由圆点隔开，比如 195.58.189.20。每个部分都是一个字节，即一个 IP 地址的字节长度为 32 位。

在分类的 IP 寻址中，IP 地址空间分为三类（从 A 类到 C 类）。通过将 IP 地址分成逻辑上的两部分——子网地址和主机号，来决定一个 IP 地址属于哪一类。子网地址用于指定网络，主机号用于在指定的网络中进行设备编址。

1. A 类

A 类 IP 地址以位序列 0-1-... 开始，例如 IP 地址的范围在 0.x.x.x ~ 127.x.x.x 之间。在 A 类子网中，IP 地址的第一个字节与子网地址相对应，后 3 个字节定义该子网中的一个节点。所以 A 类网络可以有 126 个且每个网络可包含多达一千六百万台主机。

2. B 类

B 类 IP 地址以位序列 1-0-... 开始，IP 地址的范围在 128.x.x.x ~ 191.x.x.x 之间。在 B 类子网中，IP 地址的前两个字节与子网地址相对应，后两个字节定义该子网中的一个节点。这使 B

类网络的个数可以为 16382 个且每个网络可包含多达 64000 台主机。

3. C 类

C 类 IP 地址以位序列 1-1-0-... 开始，IP 地址的范围在 192. x. x. x ~ 223. x. x. x 之间。在 C 类子网中，IP 地址的前 3 个字节与子网地址相对应，最后一个字节定义该子网中的一个节点。这使 C 类网络的个数可以为两百万个且每个网络可包含多达 254 台主机。

预留的 IP 地址：

- 10. 0. 0. 0 -10. 255. 255. 255；
- 172. 16. 0. 0 -172. 31. 255. 255；
- 192. 168. 0. 0 -192. 168. 255. 255。

上述范围的地址是预留给私有网络的，所以在该范围中的 IP 地址无法在因特网上进行路由。

子网掩码：

划分子网这种技术可以将网络（A、B、C 三类）中的主机地址空间的一部分留给子网，例如，B 类网络的地址空间可以划分成任意多个更小的子网。子网掩码把 IP 地址划分成子网地址和主机号两部分，它的结构和 IP 地址（32 位或 4 字节）相同，把子网掩码的每个位正确地分配给与 IP 地址相关的位。只有子网地址相同的主机之间才可以直接相互通信，如果主机的子网地址不同，尽管它们可能处在同一物理网络中，也无法进行通信。IP 地址分类总览见表 9-9。

表 9-9 IP 地址分类总览

类 型	子网掩码	IP 地址的范围	可利用地址数
A	255. 0. 0. 0	0. x. x. x ~ 127. x. x. x	16. 000. 000
B	255. 255. 0. 0	128. x. x. x ~ 191. x. x. x	64. 000
C	255. 255. 255. 0	192. x. x. x ~ 223. x. x. x	254

第十章　西门子PLC远程访问诊断方案

随着互联网络的发展，越来越多的用户（特别是 OEM 的用户）希望能够通过互联网络对所售出的产品进行诊断和维护，这样可以减少维护工程师到现场的时间和费用，不仅节约大量的人力和物力，同时也能为客户提供更为快捷的服务，减少了客户的损失。因此，远程诊断和服务是客户迫切需要解决的问题。

这里我们提出几种适用于西门子 PLC 远程访问的方案供大家讨论。

第一节　基于 Modem 拨号的 TeleService

该方案实际上是西门子 PLC 远程访问的标准配置，即工程师站（ES）和远程的 PLC 站之间是通过 Modem 拨号进行连接的，这样，只要在两端各放置一个 Modem，通过 TS-Adapter 连接到 PLC CPU 的 MPI 口，需要时可以进行拨号连接，通过 MPI 进行远程访问。网络配置图如图 10-1 所示。

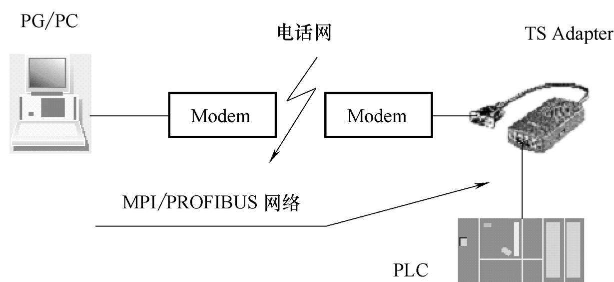


图 10-1　基于 Modem 拨号的 TeleService 网络配置图

该方案需要的软/硬件包括：

硬件：两根电话线，两个串口 Modem，一个 TS-Adapter。

软件：西门子 TeleService 软件（STEP 7 软件在本文中是默认必需的，不再单独提及）。

具体的实现方法并不复杂，操作步骤用户可以参考《西门子工业网络通信指南（上册）》一书。

这种方案的优点在于配置简单，价格便宜，无需额外的硬件卡件，如 PC 上只需要有串口，PLC 站则只需要 CPU 上的 MPI（或 PROFIBUS）口即可。

但该方案的缺点在于连接速度受限，只是拨号上网的速度，容易出现连接中断的现象。而且拨号上网的方式目前已经逐步被宽带所取代。

第二节　基于互联网的 TeleService

一、有线连接方式

在互联网上想要访问到某一个设备就需要知道该设备的 IP 地址，而该设备想要被访问也需要有一个 IP 地址，即在整个互联网上，要想访问到某一个 PLC 站，就需要该站有一个在互联网上能够被访问到的 IP 地址。

互联网上的 IP 地址一般有两种，即固定（静态）IP 地址和动态 IP 地址。

IP 地址需要向当地的 ISP 申请得到。固定（静态）IP 地址由于资源有限，因而申请和使用的费用较高，比如申请到一个端口大概是 5000 元，而固定（静态）IP 地址使用费用大概是 20000 元/月（非官方报价，仅供参考），为每个 PLC 站申请一个固定（静态）IP 地址显然是不可能的。因而靠固定（静态）IP 地址进行大量 PLC 设备的远程访问显然是不经济的（当然，这

种方式也有其应用的环境，比如实时监控）。

相比之下使用动态 IP 地址的互联网接入方式就显得较为实际。例如目前国内较为流行的 ADSL 宽带接入互联网方式，我们重点讨论的也是这种方式。

首先我们介绍一下虚拟专用网络（VPN）。

虚拟专用网络（VPN）是专用网络的扩展，它包括的链接跨 Internet 这样的共享或公用网络。使用 VPN，可以用模拟点对点专用链接的方式通过共享或公用网络在两台计算机之间传送数据。既将一些相互连接的设备组成一个虚拟的专用网络来管理。这样，对于每一个 PLC 站，都可以把它们和工程师站（ES）建立一个 VPN，从而使用工业以太网来对 PLC 站进行访问。

1. VPN 连接的建立

VPN 建立有两种形式：

1）远程用户连接：远程用户直接连接到 VPN 服务器，通过 VPN 服务器可以访问 VPN 服务器或 VPN 服务器所连接的整个网络，当然在连接的时候客户必须向服务器验证自己的身份。连接 VPN 服务器如图 10-2 所示。

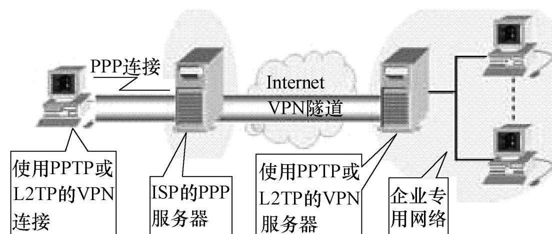


图 10-2 连接 VPN 服务器

2）路由器到路由器的连接：与上面的连接方式不同，这种 VPN 连接是通过路由器与路由器之间建立的。当然使用路由器专用的客户端软件也可以实现客户机同路由器之间直接建立 VPN 连接。通过路由器建立 VPN 连接如图 10-3 所示。

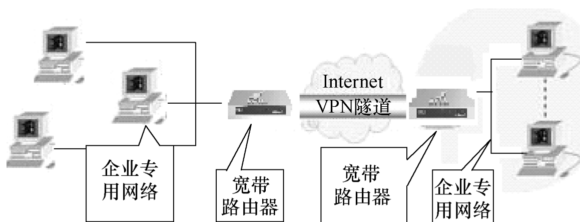


图 10-3 通过路由器建立 VPN 连接

对于远程用户直接连接到 VPN 服务器的方式比较适用于用户登录企业内部网络的应用，企业员工无论在什么地方总可以通过互联网登录到公司总部的服务器，访问企业内部网络，但对于远程诊断功能似乎有点兴师动众了，因为远程诊断并不需要企业建立一个大型的服务器来管理这些设备，只是在某一设备出了问题时才需要建立临时的连接，之后该连接可以中断，因而相比之下，在路由器之间建立 VPN 连接显得更为灵活和简便，而且投资小，无需进行 VPN 服务器等固定资产的投入，更为经济实用。

至于以太网的接入方式，目前国内比较流行的是 ADSL，用户只需向当地的电信部门申请即可，而且费用和带宽可以灵活选择，例如申请 1 兆带宽的 ADSL，选择包月上网，费用是 150 元/月，也可以选择 20 小时上网，费用为 24.5 元，十分便宜。

下面我们通过一个实际的例子对该方式进行说明。

ADSL Tele Service 配置图如图 10-4 所示。

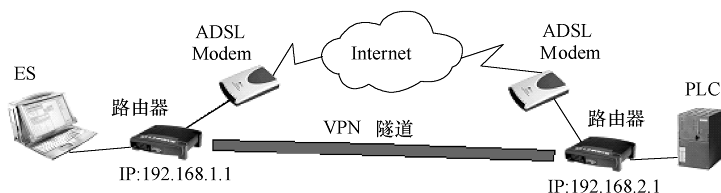


图 10-4 ADSL Tele Service 配置图

从图 10-4 中可以看到所需的硬件：两根电话线，两个 ADSL 的 Modem ，两个宽带路由器，一个工程师站（ES），一个 PLC 站（带以太网 CP 卡）。

软件则除了 STEP 7 以外没有任何额外的要求。

对于有线电话的用户申请 ADSL 服务后会得到自己的账户信息，即用户名和密码，ADSL 的设备一般由 ISP 提供。

路由器应该选择支持宽带和 VPN 功能的，在本文中我们选择了 Linksys 的一款型号为 BEFSX41 的路由器。该款路由器有 1 个 Internet 口，用于连接 ADSL Modem，4 个普通交换机的接口，用于连接本地局域网设备，如 ES 站、PLC 站等。

2. 路由器的配置

1）首先需要对两个路由器分别进行配置：将网线连接至路由器的局域网口，在 IE 浏览器中输入路由器的默认出厂设置的 IP 地址：192. 168. 1. 1 ，键入用户名和密码（默认均为“admin”）即可进入路由器的配置界面（见图 10-5）。



图 10-5 路由器的配置界面

2）在“Setup→Basic Setup→Internet”下，选择以太网连接类型：PPPoE ，用户名和密码是用户所申请的 ADSL 的用户名和密码，并且选择“Keep Alive”选项（见图 10-6）。这样，路由器即可自动通过 ADSL 的账户登录互联网。

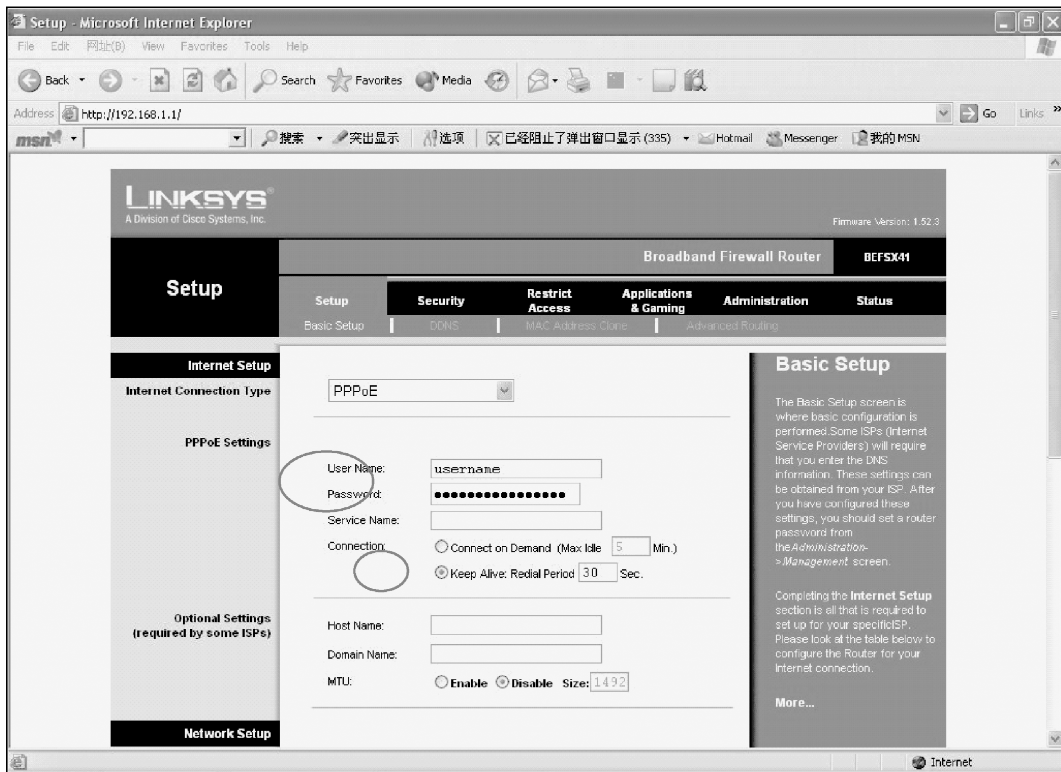


图 10-6 路由器的 ADSL 登录设置

3）对于本地网段的设置，可以设置其中一个路由器的（以下简称 R1）IP 地址为 192. 168. 1. 1，本地局域网 IP 地址池为从 192. 168. 1. 100 开始的 50 个地址，即 192. 168. 1. 100 ~ 192. 168. 1. 149，子网均为掩码 255. 255. 255. 0。选择使能本地的 DHCP Server。设置完成后注意“Save Setting”。

路由器的本地网络设置如图 10-7 所示。

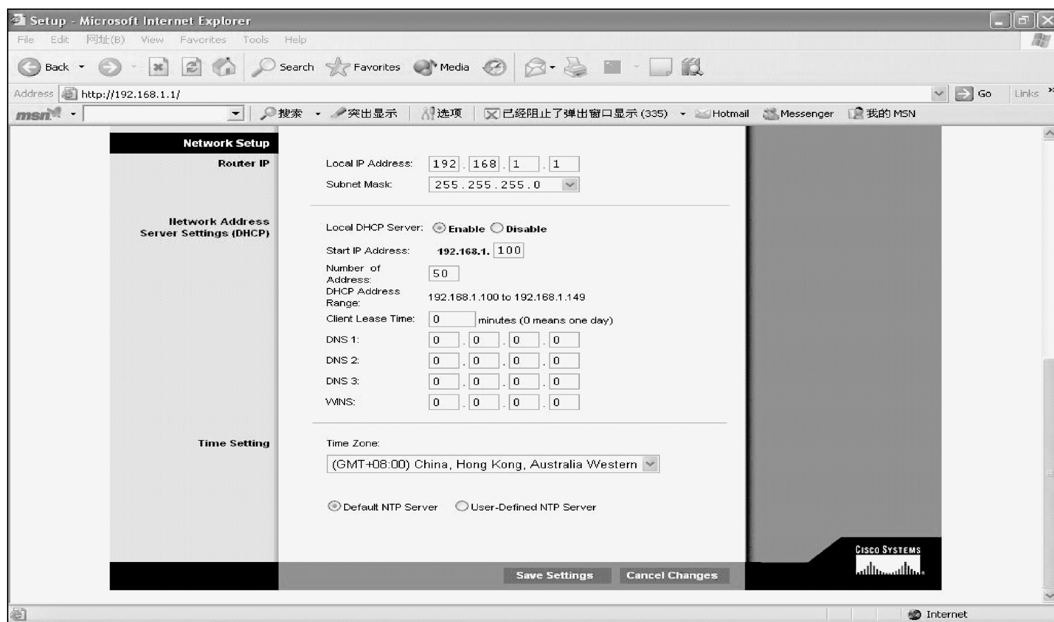


图 10-7　路由器的本地网络设置

在另外一个路由器（以下简称 R2）上的设置是一样的，只是 R2 的 PPPoE 设置为第二个 ADSL 的账户的用户名和密码，且可以将 R2 的 IP 地址设置为 192. 168. 2. 1，地址池为从 192. 168. 2. 100 开始的 50 个地址，即 192. 168. 2. 100 ~ 192. 168. 2. 149，子网均为掩码 255. 255. 255. 0 。同样可以选择使能 R2 本地的 DHCP Server。设置完成后注意“Save Setting”。

4）由于通过 ADSL 登录互联网后每次得到的 IP 地址为动态 IP 地址，因而需要使用 DDNS（动态域名服务）来对路由器的 IP 地址进行解析，这可以通过在 DDNS 服务器上注册得到。由于 Linksys 产品可以支持“PeanutHull”域名服务器，因而选择在该服务器上申请域名。这里我们使用 slc010 作为注册名称申请到两个域名：slcbj01. vicp. net 和 slcbj02. vicp. net。

打开“Setup→Basic Setup→DDNS”，将注册时域名时的用户名和密码也添加在 DDNS 参数设置中。如注册时所用的名称为：slc010（见图 10-8）。R2 可以使用相同的用户名，但最好重新申请一个不同的名字。

5）接下来设置 VPN 的连接。

打开“Security→VPN”，选择使能 VPN Tunnel ，设置名称为 VPN ，R1 的本地网地址为 192. 168. 1. 0 网段，子网掩码 255. 255. 255. 0（见图 10-9）。

相对于 R2 来讲，R1 的“Remote Security Group”是指 R2 的网段地址，即：192. 168. 2. 0，子网掩码 255. 255. 255. 0。设定动态域名及数据加密如图 10-10 所示。

而对于“Remote Security Gateways”选项来讲，这里 R1 选择的是“FQDN”，而 R2 选择“Any”即可，这样连接 VPN 时，由 R1 作为 Client 端来连接 R2。

“Fully-Qualified Domain”中的域名为“slcbj02. vicp. net”。该域名即为申请到的 DDNS 的动态域名。R2 的域名为“slcbj01. vicp. net”，与 R1 不同。

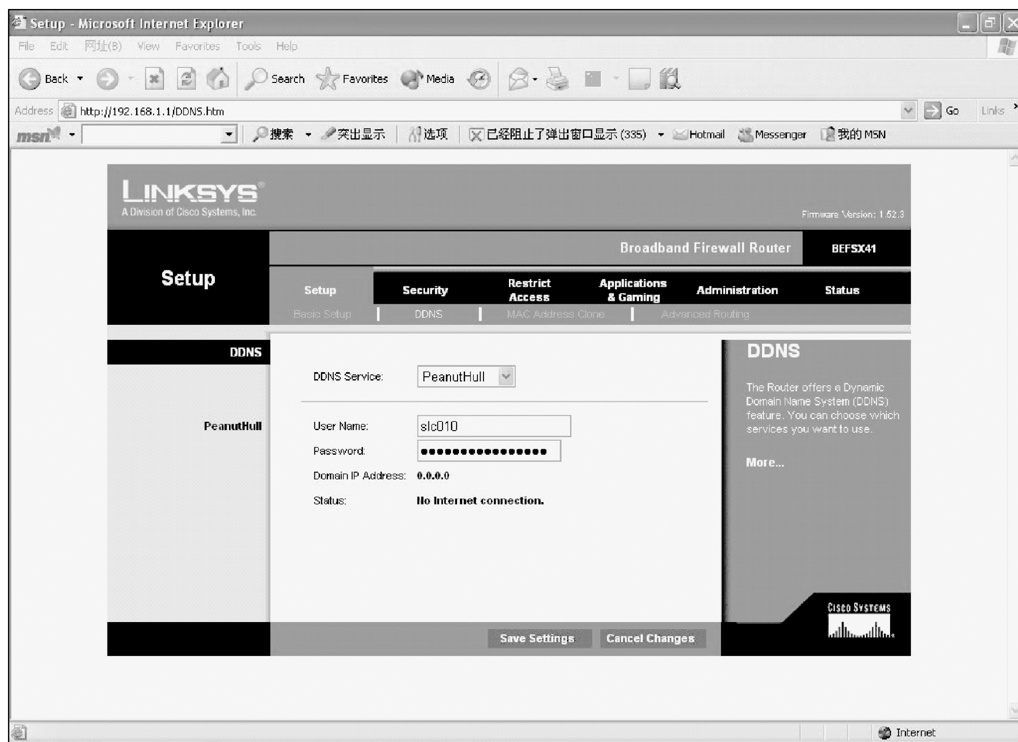


图 10-8　DDNS 的设定

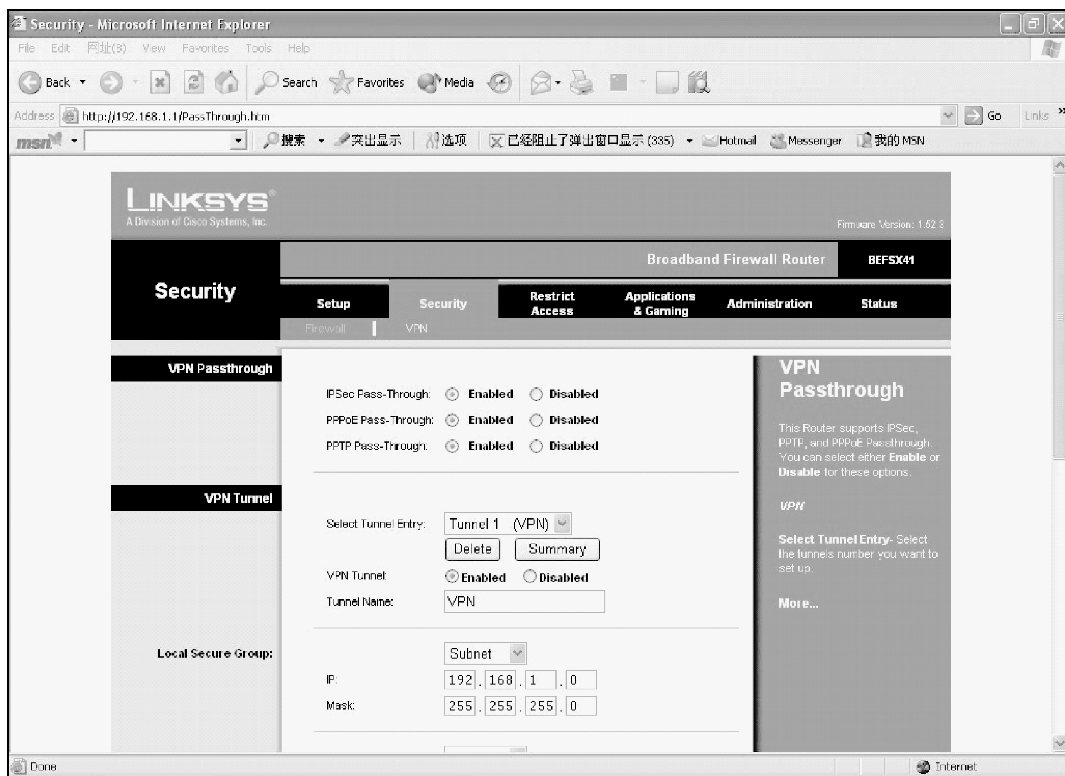


图 10-9　添加 VPN 通道

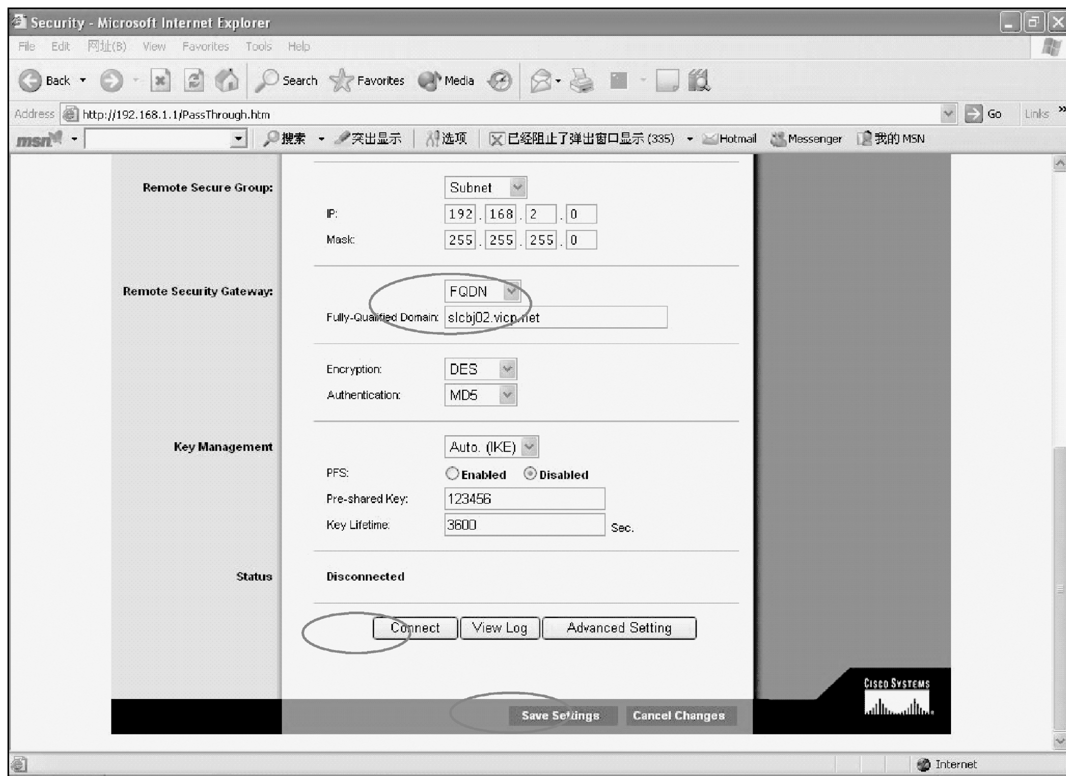


图 10-10　设定动态域名及数据加密

对于数据密钥的设定，R1 和 R2 的设定必须相同。“Advanced Setting”也必须相同，且“Pre-shared Key”不能为空。数据密钥的高级设置如图 10-11 所示。

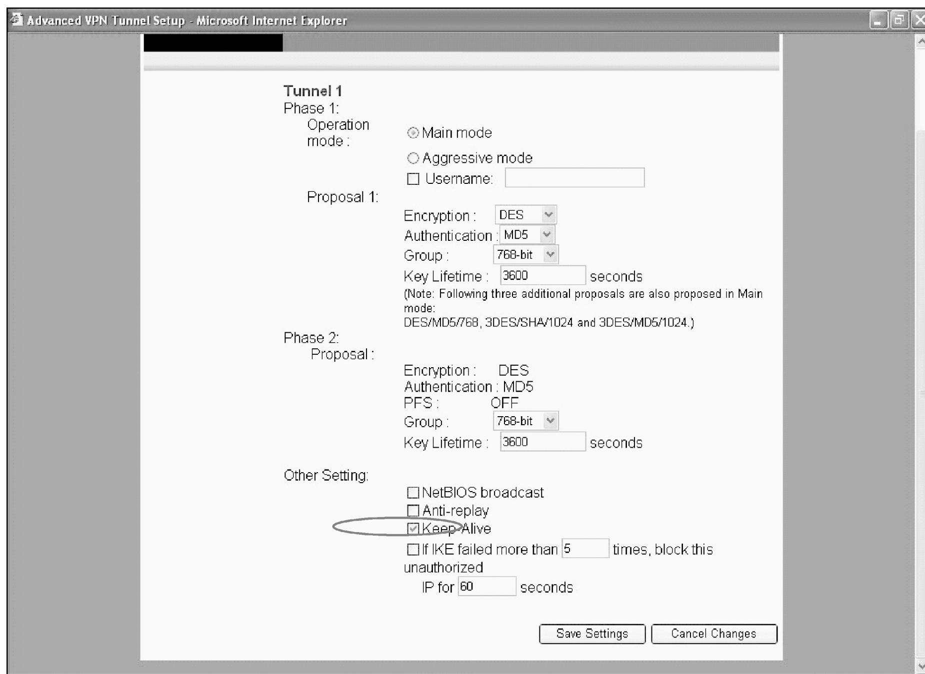


图 10-11　数据密钥的高级设置

6）当“Save Setting”后，两个路由器可以自动拨号，通过各自的 ADSL 账号连接到互联网，且 R1 自动连接 R2，建立 VPN 通道。可以通过状态检测来观察连接的情况。

检查连接状态如图 10-12 所示。



图 10-12　检查连接状态

3. PLC 站的组态

首先要对 PLC 站进行组态。PLC 站的以太网参数设置如图 10-13 所示，设定 PLC 站以太网的 IP 地址，由于 PLC 站连接在 R2 后面，因而它的 IP 地址应该设定在 192.168.2.100 ~ 192.168.2.149 之间，且选择“Use router”选项，添加路由器 R2 的 IP 地址 192.168.2.1 。

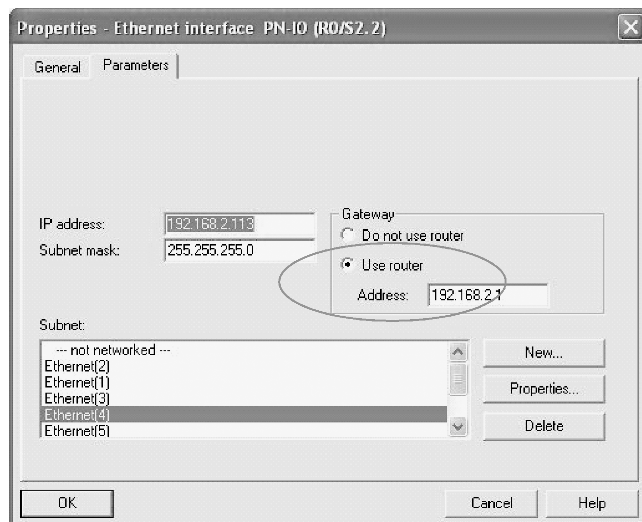


图 10-13　PLC 站的以太网参数设置

将参数下载保存在 PLC 站后，将 PLC 站连接在 R2 后。当 VPN 的连接建立时，连接在 R1 后的 ES 站可以通过 STEP7（包括 WinCC 等）对远端的 PLC 站进行远程访问。STEP7/WinCC 通过 ADSL 访问远程 PLC 站的在线、运行画面如图 10-14 所示。

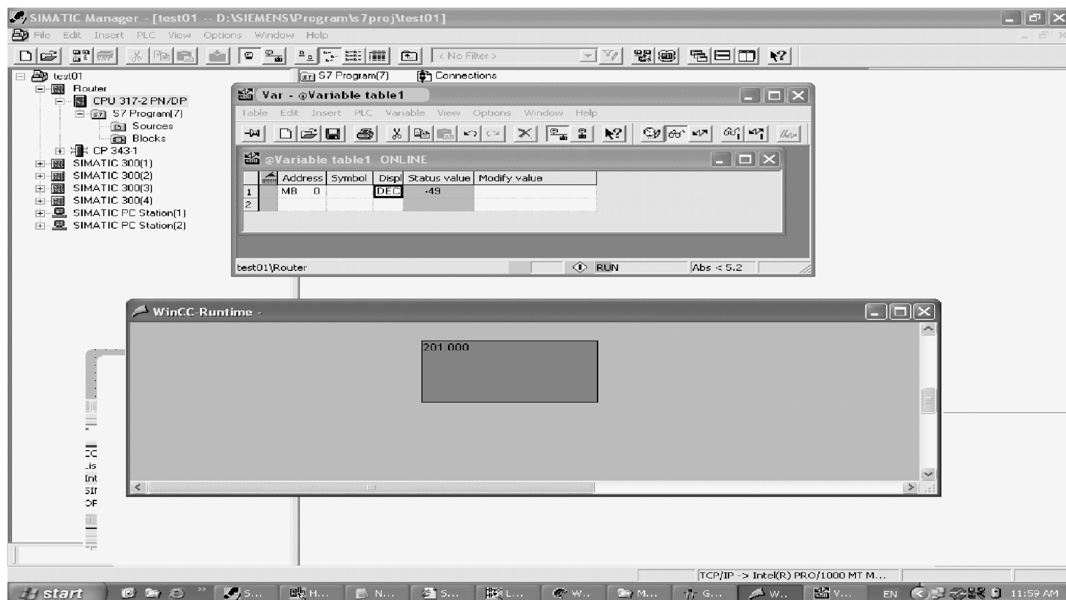


图 10-14　STEP7/WinCC 通过 ADSL 访问远程 PLC 站的在线、运行画面

二、无线方式（CDMA/GPRS）建立 VPN

在某些场合可能没有电话线，或者如果用户希望随时随地都可以对设备进行诊断，这样通过有线电话拨 ADSL 建立 VPN 的方式则会受到限制，此时用户可以考虑采用无线通信的方式建立 VPN。

这里的无线通信 VPN 的方式需要通过支持无线通信（如 GPRS/CDMA ）的宽带路由器来完成，通过无线宽带路由器建立 VPN 如图 10-15 所示。

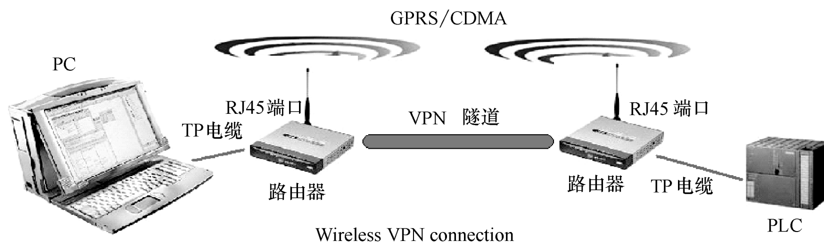


图 10-15　通过无线宽带路由器建立 VPN

在图 10-15 中，我们可以通过两个无线路由器来建立一个 VPN 的通道，此时，将支持 GPRS 或 CDMA 的 SIM 卡分别插在两个 Router 中（SIM 卡开通数据业务须向当地的移动通信部门申请），这样，通过设置该 Router 就可以像有线 ADSL 一样在两个 Router 之间建立一个 VPN 通道，从而实现远程连接。通过无线网卡和宽带路由器建立 VPN 如图 10-16 所示。

在图 10-16 中，我们可以通过一个无线网卡和路由器来建立 VPN 的通道，此时，将支持 GPRS 或 CDMA 的 SIM 卡分别插在网卡和 Router 中，通过该 Router 制造商提供的 VPN 客户端软件，可以将该移动 PC 同无线 Router 建立 VPN 的连接，从而实现在没有电话线或工程师站是可移动的情况下对某固定设备进行远程诊断。

CDMA 无线上网最高速率可达 153.6kbit/s，稳定状态下的速率可在 70～80kbit/s 左右，是普通拨号上网的 3 倍以上。GPRS 上网的峰值速率为 115.2kbit/s，平均上网速率在 20～30kbit/s。

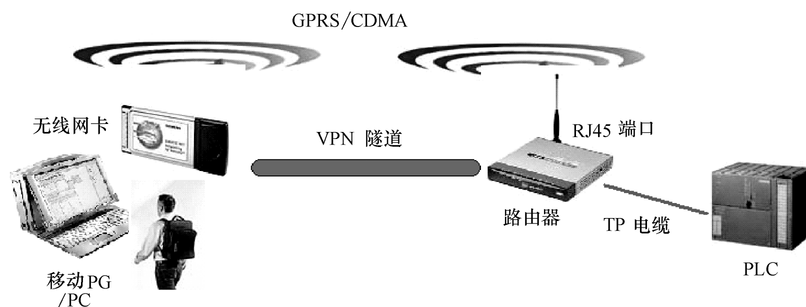
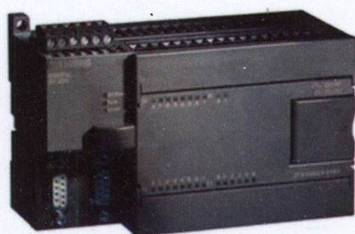


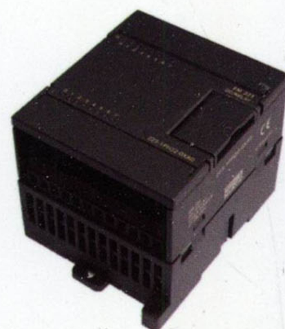
图 10-16 通过无线网卡和宽带路由器建立 VPN

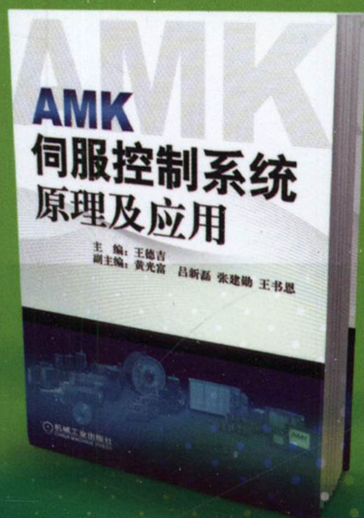
上述两者提供数据业务的方式不同，CDMA 传输速率依赖无线环境程度不大；而 GPRS 的数据业务与话音业务共用同一信道，如果网络用户数量增加到一定程度，可导致每个 GPRS 用户使用的带宽进一步降低，因而，基于 CDMA 网络的无线上网业务在速度和稳定性等方面优于 GPRS。

当然，这样的连接速度是无法与有线 ADSL 相比的，但为了满足特定环境下的特定用户的需求，不失为一种解决方案。

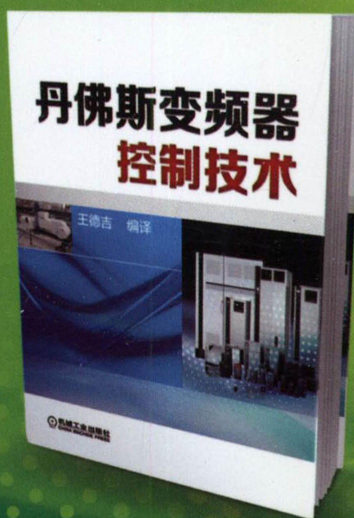
以上，我们讨论了对 PLC 站进行远程访问的几种方式，可以说各种方式都有其应用的场合，用户可以根据实际情况进行选择，也可以混合使用。比如，如果 CP343-1 或 CP443-1 出现问题而无法通信，基于互联网的远程诊断功能就要受限，这时只能通过 TS-Adapter 直接连接 CPU 来进行远程诊断，总之，我们讲远程诊断的宗旨就是能够以最低的成本完成对 PLC 设备的远程诊断和维护。







书号：40188
定价：35.00



书号：46400
定价：29.90



书号：46052
定价：78.00

地址：北京市百万庄大街22号
邮政编码：100037
电话服务
社服务中心：010-88361066
销售一部：010-68326294
销售二部：010-88379649
读者购书热线：010-88379203
网络服务
教材网：http://www.cmpedu.com
机工官网：http://www.cmpbook.com
机工官博：http://weibo.com/cmp1952
封面无防伪标均为盗版

ISBN 978-7-111-46052-7
策划编辑◎林春泉

ISBN 978-7-111-46052-7



定价：78.00元